**Luiz Henrique Nogueira Lorena**

# NOVAS FORMULAÇÕES E TÉCNICAS DE PRÉ-PROCESSAMENTO PARA O PROBLEMA DE PARTICIONAMENTO DE GRAFO EM CLIQUES

Tese apresentada à Universidade Federal São Paulo, para obtenção do título de Doutor em Ciência da Computação.

São José dos Campos

2019

**Luiz Henrique Nogueira Lorena**

# NOVAS FORMULAÇÕES E TÉCNICAS DE PRÉ-PROCESSAMENTO PARA O PROBLEMA DE PARTICIONAMENTO DE GRAFO EM CLIQUES

Tese apresentada à Universidade Federal São Paulo, para obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Marcos Gonçalves Quiles

Coorientador: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho

São José dos Campos

2019

Lorena, Luiz Henrique Nogueira

**Novas formulações e técnicas de pré-processamento para o problema de particionamento de grafo em cliques** / Luiz Henrique Nogueira Lorena. - São José dos Campos, 2019

xv, 91 f.

Tese (Doutorado) - Universidade Federal de São Paulo. Instituto de Ciência Ciência e Tecnologia. Programa de Pós-Graduação em Ciência da Computação

Título em inglês: New formulations and preprocessing techniques for the clique partitioning problem.

1. Agrupamento de dados em grafo. 2. Programação Linear Inteira
3. Técnicas de pré-processamento

**UNIVERSIDADE FEDERAL DE SÃO PAULO**
**INSTITUTO DE CIÊNCIA E TECNOLOGIA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

Chefe do Departamento: Prof. Dr. Carlos Marcelo Gurjão Godoy

Coordenador do curso de pós-graduação: Prof. Dr. Valério Rosset

**Luiz Henrique Nogueira Lorena**

# NOVAS FORMULAÇÕES E TÉCNICAS DE PRÉ-PROCESSAMENTO PARA O PROBLEMA DE PARTICIONAMENTO DE GRAFO EM CLIQUES

Tese apresentada à Universidade Federal São Paulo, para obtenção do título de Doutor em Ciência da Computação.

Área de Concentração: Sistemas Inteligentes

Aprovada em 7 de junho de 2019

Presidente da banca:

Prof. Dr. Marcos Gonçalves Quiles

Banca examinadora:

Prof. Dr. Carlos Henrique Costa Ribeiro

Prof. Dr. Filipe Alves Neto Verri

Prof. Dr. Nandamudi Lankalapalli Vijaykumar

Profa. Dra. Mariá Cristina Vasconcelos Nascimento Rosset

Dedico esta tese aos meus pais, minha irmã e sobrinhos. Eles foram responsáveis pela paz nos momentos em que me encontrei incapaz de prosseguir. Eu amo vocês.

# RESUMO

O agrupamento de dados em grafo é uma técnica fundamental em análise de dados, utilizada para compreender a relação entre as entidades de um conjunto de dados. Neste trabalho, estudou-se um problema de agrupamento de dados em grafo em particular denominado Particionamento de Grafo em Cliques. Neste problema, o objetivo é particionar os vértices de um grafo completo ponderado, de forma que o valor da soma do peso das arestas dentro dos grupos obtidos seja máximo. Os autores que sugeriram esse problema criaram uma formulação de programação linear inteira para solucioná-lo. Essa formulação, entretanto, cria modelos matemáticos com um elevado número de variáveis e restrições, e que demandam um elevado esforço computacional, em termos de tempo e memória, para serem solucionados. Os autores, no entanto, demonstraram empiricamente que uma grande número de restrições inseridas nos modelos matemáticos são redundantes e podem ser desconsideradas. Apesar disso, não realizaram um estudo para tentar identificar o porquê desse comportamento. Apenas recentemente, foram realizados estudos que tentam identificar a redundância dessa formulação. Nesses estudos, os autores conseguiram identificaram parte da redundância e propuseram formulações que evitam que ela seja inserida no modelo matemático, criando um modelo que demandam menos esforço computacional para ser solucionado. Uma outra abordagem utilizada na literatura para tentar mitigar os problemas da formulação original é diminuir o tamanho do grafo do problema, através de uma técnica de pré-processamento. Um grafo menor dá origem a um modelo matemático com menos variáveis e restrições. Esta tese seguiu os princípios desses trabalhos para propor novas formulações e técnicas de pré-processamento. Os experimentos computacionais mostram que as técnicas propostas neste trabalho dão origem à modelos matemáticos mais compactos, e que demandam menos esforço computacional para serem solucionados do que os modelos criados a partir das formulações propostas na literatura.

**Palavras-chave:** Agrupamento de dados em grafo, Programação linear inteira, Técnicas de pré-processamento.

## ABSTRACT

Graph clustering is a fundamental technique in data analysis, used to understand the relation between the entities of a dataset. In this work, we have studied a graph clustering problem called Clique Partitioning Problem. In this problem, the objective is to partition the vertices of a weighted complete graph, such that the value of the sum of the edge's weight within the obtained groups is maximum. The authors who suggested this problem proposed a linear integer programming formulation to solve it. This formulation, however, creates mathematical models with a high number of variables and constraints, which require a high computational effort, in terms of time and memory, to be solved. The authors, however, have empirically demonstrated that a large number of constraints inserted in mathematical models are redundant and can be disregarded. Despite this, the authors did not identified the reason for such behavior. Recent studies try to identify the redundancy of this formulation and prevent it to be inserted into the mathematical model. Such models require less computational effort to be solved. Another approach used in the literature, to attenuate the issue of the original formulation, is to reduce the size of the graph instance that represents the problem through a preprocessing technique. A smaller graph gives rise to a mathematical model with fewer variables and constraints. This thesis followed the concepts presented in these works to propose new formulations and preprocessing techniques. The computational experiments show that the techniques proposed in this work create mathematical models that are more compact, and which require less computational effort to be solved than the models created from the formulations proposed in the literature.



**Keywords:** Graph clustering, Integer linear programming, Preprocessing techniques

# AGRADECIMENTOS

Essa tese não teria se tornado realidade se não fossem as oportunidades, o apoio, os incentivos e a disponibilidade de diversas pessoas.

Gostaria de agradecer meus orientadores pela oportunidade que me deram de realizar essa tese, a disponibilidade para esclarecer minhas dúvidas, ao conhecimento que me transmitiram, as contribuições, e as palavras de incentivo. Faço um agradecimento em especial, ao professor Dr. Marcos Gonçalves Quiles, pela paciência e por dar apoio em questões que foram muito além dos problemas analisados nessa tese. Suas palavras foram essenciais para me dar forças para seguir em frente. Agradeço ao professor Dr. André Carlos Ponce de Leon Ferreira de Carvalho por ter aceitado o convite para trabalharmos juntos novamente, este trabalho não seria possível sem a oportunidade que me foi dada quando você me orientou pela primeira vez no mestrado.

Agradeço à UNIFESP pela oportunidade que me foi dada para realizar essa tese. Estendo meus agradecimentos aos docentes, funcionários e discentes.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de estudos de Doutorado (CAPES-DS).

Por último, gostaria de agradecer meus pais pelo apoio incondicional, incentivo, paciência e ajuda nos obstáculos que surgiram ao longo dessa caminhada. Esse trabalho não seria possível sem vocês.

SUMÁRIO

1 INTRODUÇÃO 1
1.1 Objetivos 3
1.2 Contribuições 3
1.3 Organização do texto 4
2 CONCEITOS BÁSICOS 6
2.1 Definição de um grafo 6
2.2 Representação de um grafo 6
2.3 Grau de um vértice e vizinhança 6
2.4 Subgrafos 7
2.5 Conectividade 7
2.6 Grafo completo e cliques 8
3 PARTICIONAMENTO DE GRAFO EM CLIQUES 9
3.1 Aplicações do PGC e trabalhos relacionados 12
3.2 Formulações PLI que criam modelos matemáticos mais compactos para o PGC 13
4 PGC E DETECÇÃO DE COMUNIDADES 16
4.1 Detecção de comunidades por maximização da modularidade 16
4.2 Maximização de modularidade via PLI 17
4.3 Técnicas de pré-processamento 18
4.4 Filtro LCCFilter 20
4.5 Filtro BCCFilter 23
4.6 Resultados experimentais 26
4.6.1 Desempenho dos filtros *LCCFilter* e *BCCFilter* 27
4.6.2 Solução dos modelos matemáticos 30
4.7 Conclusões 32
5 PGC E AGRUPAMENTO DE DADOS QUALITATIVOS 34
5.1 Agrupamento de dados qualitativos via PLI 35
5.2 Uma nova formulação PLI para o agrupamento de dados qualitativos 37
5.3 Resultados experimentais 41
5.3.1 Eliminação de restrições e tempo computacional 43
5.3.2 Eficiência de espaço 45
5.3.3 Robustez 47
5.3.4 Significância estatística 48
5.4 Conclusões 48
6 PGC E PROBLEMA DE EDIÇÃO DE CLUSTERS 50
6.1 O problema de Edição de Clusters 50
6.2 Edição de clusters via PLI 51
6.3 Uma nova formulação para o PEC 52

6.4 Resultados experimentais 54
6.4.1 Experimentos com grafos LFR 55
6.4.2 Experimentos com grafos aleatórios não-ponderados 57
6.5 Conclusões 60
7 CONCLUSÃO 61
7.1 Trabalhos futuros 62
7.1.1 Relaxação de Programação Linear do PGC 63
7.1.2 Propor uma formulação PGC mais compacta para modularidade 63
7.1.3 Propor uma formulação mais compacta para o PGC com restrições de capacidade 64
7.1.4 Propor uma formulação mais compacta para o problema de Agregação de *Rankings* 65
REFERÊNCIAS BIBLIOGRÁFICAS 67

# LISTA DE FIGURAS


Figura 1 Exemplos de subgrafos. 7
Figura 2 Exemplo de um grafo biconexo, não biconexo e de componentes biconexos. 8
Figura 3 Exemplo de um grafo completo. 8
Figura 4 Representação gráfica do grafo completo de entrada do PGC e de uma solução possível para essa instância. 9
Figura 5 Grafo de citações criado a partir das referências relacionadas ao PGC, organizado em ordem cronológica. Quadrados representam heurísticas e círculos representam métodos exatos. 14
Figura 6 Análise de um grupo de uma partição ótima. 15
Figura 7 Triângulo correspondente à restrição de transitividade 4 e respectiva cláusula condicional. 15
Figura 8 Partição do grafo *Zachary's karate club* [89] e sua representação reduzida. 19
Figura 9 Tipos de subgrafos periféricos pré-processados por Arenas et al. [9]. 20
Figura 10 Exemplo de grafo com quatro cliques $K_s$ periféricos. 21
Figura 11 Identificando cliques $K_s$ através do LCC dos vértices. 22
Figura 12 Exemplo ilustrando cada etapa do Algoritmo 2. 23
Figura 13 Identificando os bicomponentes do grafo $G$. Vértices que são pontos de articulação em $G$ são representados como um quadrado. 24
Figura 14 Identificação de outros tipos de subgrafos periféricos através da decomposição de $G$ em bicomponentes. Vértices que são pontos de articulação em $G$ são representados como um quadrado. 24
Figura 15 Exemplo ilustrando cada etapa do Algoritmo 3. 26
Figura 16 Partição ótima do conjunto de dados 6 (netscience). 28
Figura 17 Percentual de redução do tamanho dos grafos obtidos pelos filtros em relação ao tamanho do grafo original. 29
Figura 18 Percentual de redução do tamanho dos modelos matemáticos criados a partir dos grafos resultantes da aplicação dos filtros *LCCFilter* e *BCCFilter*. 31
Figura 19 Percentual de redução em tempo e memória obtido pelos modelos matemáticos criados a partir dos grafos resultantes da aplicação dos filtros *LCCFilter* e *BCCFilter*. 32

Figura 20 Grafo que representa o exemplo da Tabela 6 e uma possível solução. Arestas com peso negativo foram representadas através de uma linha pontilhada. 36
Figura 21 Exemplo em que todas arestas em $E_{AB}$ possuem peso negativo. 38
Figura 22 Exemplo em que a soma do peso das arestas em $E_{AB}$ possui valor negativo. 39
Figura 23 Tipos de triângulos que podem ser encontrados em $K_n$. 40
Figura 24 Metodologia utilizada nos experimentos computacionais. 42
Figura 25 Percentual de restrições eliminadas e *speedup* obtido pelos modelos matemáticos criados a partir das formulações $PGC_{R1}$, $PGC_{R2}$, $PGC_{R3}$ frente ao modelo criado a partir da formulação $PGC$. 45
Figura 26 Percentual de redução do consumo de memória dos modelos matemáticos criados a partir das formulações $PGC_{R1}$, $PGC_{R2}$, $PGC_{R3}$ frente ao modelo criado a partir da formulação $PGC$. 46
Figura 27 Percentual de restrições de transitividades eliminadas do modelo matemático a medida que se aumenta o número de arestas com peso positivo no grafo completo do problema. 47
Figura 28 Exemplo de Edição de *Clusters*. Linhas sólidas, tracejadas e pontilhadas representam, respectivamente, arestas que foram mantidas, removidas e inseridas. 50
Figura 29 Adaptando o PEC ao contexto do PGC. 51
Figura 30 Tipos de triângulos que podem existir em $K_n$. 53
Figura 31 Possíveis configurações para o peso das arestas de triângulos do tipo $T2$. 54
Figura 32 Partição ótima do conjunto de dados Zachary's karate club [89]. A aresta $\{33, 34\}$ foi destacada em vermelho. 64

## LISTA DE TABELAS

Tabela 1 Conjuntos de dados utilizados nos experimentos. 27
Tabela 2 Tempo computational (segundos), tipo e quantidade de subgrafos periféricos filtrados por *LCCFilter* e *BCCFilter*. 28
Tabela 3 Tamanho dos grafos após aplicar os filtros *LCCFilter* e *BCCFilter*. 29

Tabela 4 Modularidade e tamanho dos modelos matemáticos criados a partir do grafo original e dos grafos filtrados por *LCCFilter* e *BCCFilter*. 30
Tabela 5 Tempo computacional (segundos) e consumo de memória (*megabytes*) dos modelos PLI criados a partir do grafo original e dos grafos filtrados por *LCCFilter* e *BCCFilter*. 32
Tabela 6 Conjunto de dados exemplo. 35
Tabela 7 Conjuntos de dados utilizados nos experimentos. 43
Tabela 8 Valor da função objetivo, número de restrições dos modelos matemáticos e tempo computacional (segundos) necessário para solucioná-los. 44
Tabela 9 Consumo de memória, em *megabytes*, dos modelos matemáticos. 46
Tabela 10 Desempenho obtido pelo *solver* na resolução dos modelos criados a partir das formulações $\mathrm{PEC}_{\mathrm{ILP}}$ e $\mathrm{PECR}_{\mathrm{ILP}}$ nos grafos LFR. 56
Tabela 11 Desempenho obtido pelo *solver* na resolução dos modelos criados a partir das formulações $\mathrm{PEC}_{\mathrm{ILP}}$ e $\mathrm{PECR}_{\mathrm{ILP}}$ nos grafos aleatórios com até 300 vértices. 58
Tabela 12 Desempenho obtido pelo *solver* na resolução dos modelos criados a partir das formulações $\mathrm{PEC}_{\mathrm{ILP}}$ e $\mathrm{PECR}_{\mathrm{ILP}}$ nos grafos aleatórios com até 2000 vértices. 59

## ACRÔNIMOS

**PGC** Particionamento de Grafo em Cliques

**PLI** Programação Linear Inteira

**PEC** Problema de Edição de *Clusters*

## LISTA DE SÍMBOLOS

$G$ Grafo

$V(G)$ Conjunto de vértices de um grafo $G$

$E(G)$ Conjunto de arestas de um grafo $G$

| | |
|---|---|
| $n$ | Número de vértices de um grafo |
| $m$ | Número de arestas de um grafo |
| $\{i, j\}$ | aresta entre os vértices $i$ e $j$ |
| $A$ | Matriz de adjacências de um grafo |
| $a_{ij}$ | Elemento localizado na $i$-ézima linha e $j$-ésima coluna da matriz de adjacências de um grafo |
| $W$ | Matriz de pesos de um grafo |
| $w_{ij}$ | Elemento localizado na $i$-ézima linha e $j$-ésima coluna da matriz de pesos de um grafo |
| $N_i$ | Conjunto formado por todos os vizinhos do vértice $i$ |
| $d_i$ | Grau do vértice $i$ |
| $s_i$ | Força do vértice $i$ |
| $\mathcal{P}$ | Partição de um grafo |
| $K_n$ | Grafo completo composto por $n$ vértices |

# 1 INTRODUÇÃO

Organizar dados em grupos, também conhecido como agrupamento de dados, é fundamental para muitas aplicações em análise de dados. É frequentemente utilizado para aprender e compreender a relação entre as entidades que compõem um conjunto de dados. Dada a importância do agrupamento de dados, diversas técnicas e variantes desse problema foram propostas na literatura [51, 2].

Entre as diversas técnicas, as baseadas em distância ou similaridade [51], por exemplo, são frequentemente empregadas devido à simplicidade de implementação e facilidade de adaptação a diversos contextos. Assim como as técnicas, diversas variantes do problema de agrupamento também foram propostas na literatura [2]. A variante mais comumente encontrada na literatura tem como objetivo definir grupos disjuntos que possuam o maior valor de similaridade entre seus membros. Existe, entretanto, variantes que consideram grupos com sobreposição ou que considerem restrições para o tamanho dos grupos, por exemplo.

Este trabalho aborda uma variante específica do problema de agrupamento de dados, em que o problema é modelado através de um grafo e o objetivo é agrupar as entidades de um conjunto de dados em grupos disjuntos com maior valor de similaridade entre os seus membros. Essa variante é interessante pois é facilmente adaptável a diversos contextos, sendo necessário apenas definir uma função de similaridade adequada ao contexto do problema analisado. Esse tipo de abordagem tem sido utilizada com sucesso em diferentes campos da ciência, como ciência da computação, física, química, biologia e sociologia [81, 4, 68].

Grotschel e Wakabayashi [42] foram os primeiros a estudarem a resolução dessa variante do problema de agrupamento de dados de forma exata, modelando-o como um problema de particionamento de um grafo denominado Particionamento de Grafos em Cliques (PGC). No PGC, uma instância do problema de agrupamento de dados é representada através de um grafo completo, em que os vértices representam as entidades a serem agrupadas e o peso das arestas representam a similaridade entre as entidades. O objetivo do PGC é particionar o grafo completo ponderado de forma que sejam obtidos grupos disjuntos de vértices que possuam o maior valor de similaridade entre seus membros.

Para resolver o PGC de forma exata, Grotschel e Wakabayashi [42] propuseram uma formulação baseada em Programação Linear Inteira (PLI). A formulação é interessante pois não necessita que o número de grupos seja definido previamente, esse valor é obtido diretamente através do processo de

otimização da função objetivo do problema. Essa formulação, entretanto, gera modelos matemáticos compostos por um grande número de variáveis e restrições, e que demandam um excessivo esforço computacional para serem solucionados de forma exata.

Nesse mesmo trabalho, entretanto, os autores fizeram um estudo empírico que mostrou que um grande número de restrições previstas na formulação são redundantes. Os autores observaram que as restrições redundantes não são necessárias para resolver o problema e, portanto, podem ser desconsideradas durante a construção do modelo matemático. Os autores, entretanto, não apontaram razões específicas para explicar a redundância, apenas observaram empiricamente que um grande número de restrições eram desnecessárias. Esse tipo de análise foi realizado apenas em trabalhos mais recentes da literatura.

Um estudo pioneiro realizado por Dinh e Thai [27] conseguiu identificar situações em que uma restrição pode ser considerada redundante. Os autores criaram a partir desse estudo uma formulação que evita que essa redundância seja inserida no modelo matemático. Os experimentos apresentados pelos autores mostraram que o modelo matemático criado a partir dessa formulação foi capaz de obter a solução ótima do problema, necessitando menos tempo computacional e memória para ser solucionado de forma exata. Recentemente, Miyauchi e Sukegawa [67] estenderam a técnica de Dinh e Thai [27], identificando outras situações em que uma restrição pode ser considerada redundante, criando uma formulação que gera modelos matemáticos ainda mais compactos.

Uma outra forma de melhorar o desempenho dos modelos matemáticos criados a partir da formulação proposta em [42], explorada na literatura, é diminuir o tamanho do grafo que serve de entrada para o PGC. Uma eventual redução no número de vértices e arestas desse grafo irá proporcionar uma redução geral no número de variáveis e restrições do modelo matemático. Novamente, Dinh e Thai [27] foram os primeiros a explorarem essa abordagem para melhorar o desempenho da técnica que irá resolver o problema de forma exata. Os autores utilizaram uma técnica de pré-processamento, proposta por Arenas et al. [9], para reduzir o tamanho do grafo do problema sem prejudicar a estrutura de grupos presente no grafo original.

Os trabalhos de Dinh e Thai [27], Miyauchi e Sukegawa [67] e Arenas et al. [9] motivaram o estudo realizado nesta tese, e serviram de base para as técnicas propostas. Além disso, serviram para definir a seguinte hipótese de pesquisa para este trabalho: é possível desenvolver formulações que resultem em modelos matemáticos mais compactos que os obtidos pelas formulações apresentadas em [42, 27, 67] e técnicas de pré-processamento do grafo do problema que obtenham resultados melhores que a técnica apresentada em [9] em termos de redução do grafo do problema. Mais detalhes sobre esses trabalhos serão apresentados no próximo capítulo.

Existe um conjunto de fatores que motivou o estudo da formulação exata para o PGC:

- Estudar a redundância da formulação original proposta em [42] é um assunto que foi pouco explorado na literatura.
- A formulação é aplicada em diversos contextos da literatura.
- O estudo do problema no contexto do método exato pode ajudar na construção de novas heurísticas ou pode sugerir melhorias em uma heurística existente (novos movimentos de troca, por exemplo).
- A solução ótima obtida para instâncias de maior tamanho podem servir como *benchmark* para que heurísticas possam ser comparadas ou refinadas.

## 1.1 OBJETIVOS

O objetivo **geral** deste trabalho é:

- Desenvolver formulações PGC que resultem em modelos matemáticos mais compactos que os obtidos pelas formulações propostas anteriormente na literatura, e novas técnicas de pré-processamento para o PGC.

Este trabalho possui os seguintes objetivos **específicos**:

- Estudar e adaptar diferentes técnicas de pré-processamento e formulações que resultem em modelos matemáticos mais compactos para o PGC;
- Empregar essas técnicas em diferentes problemas em que o PGC pode ser aplicado;
- Avaliar as técnicas propostas, comparando-as com outros resultados da literatura.

## 1.2 CONTRIBUIÇÕES

Uma primeira contribuição deste trabalho é apresentada no Capítulo 4, em que o PGC é aplicado no contexto de detecção de comunidades. Duas técnicas de pré-processamento do grafo do problema PGC são propostas. Elas estendem a técnica de pré-processamento originalmente proposta por Arenas et al. [9]. Os resultados experimentais mostram que as novas técnicas superam a técnica de proposta por Arenas et al. [9], criando grafos menores. Essa redução beneficia a solução do problema PGC de forma exata. Parte dos resultados desse capítulo fizeram parte de um artigo que foi aceito para

apresentação oral e publicação [64] nos anais da conferência ICCSA 2019 (*International Conference on Computational Science and its Applications*).

No Capítulo 5, é apresentada uma formulação que dá origem a modelos matemáticos mais compactos para o PGC no contexto de agrupamento de dados qualitativos. Foi realizado um estudo computacional com o objetivo de identificar quais restrições da formulação proposta por Grotschel e Wakabayashi [42] são redundantes dentro desse contexto. Os resultados desse estudo foram utilizados para propor um nova formulação, denominada $\mathrm{PGC}_{\mathrm{R3}}$. Foram realizados experimentos para comparar os modelos matemáticos criados a partir da nova formulação, da formulação original proposta em [42] e das formulações mais recentes propostas por Miyauchi e Sukegawa [67]. Os resultados experimentais mostram que os modelos matemáticos criados a partir da formulação $\mathrm{PGC}_{\mathrm{R3}}$ consomem menos tempo computacional e memória, para serem solucionados de forma exata, do que os modelos matemáticos criados a partir das outras formulações consideradas [42, 67]. Os resultados desse capítulo fizeram parte de um artigo que foi aceito para apresentação oral e publicação [65] nos anais da conferência IJCNN 2019 (*International Joint Conference on Neural Networks*).

Finalmente, o Capítulo 6 apresenta uma formulação que cria modelos matemáticos mais compactos para o PGC aplicado ao contexto do problema de Edição de *Clusters*. Novamente, foram identificadas quais restrições da formulação proposta originalmente em [42] devem ser consideradas para esse problema. Nos experimentos computacionais, é medido o desempenho da técnica que irá resolver os modelos matemáticos criados a partir da formulação original [42] e da nova formulação. Os resultados mostram que a nova formulação conseguiu criar modelos matemáticos com um número consideravelmente menor de restrições. Essa diminuição do tamanho dos modelos fez com que a técnica empregada para resolvê-los de forma exata consumisse menos tempo computacional e memória do que os modelos matemáticos criado através da formulação original. Os resultados apresentados nesse capítulo foram apresentados no ICIC 2018 (*International Conference on Intelligent Computing*) e publicados nos anais do congresso (Lorena et al. [63]).

## 1.3 ORGANIZAÇÃO DO TEXTO

Este documento está organizado em sete capítulos. O Capítulo 1 apresenta as motivações, objetivos e principais contribuições deste trabalho. Conceitos de Teoria de Grafos e a notação que será utilizada ao longo deste trabalho são apresentados no Capítulo 2. O Capítulo 3 faz uma introdução ao problema de particionamento de grafo em cliques e apresenta os principais trabalhos relacionados. No Capítulo 4 serão apresentadas as primeiras contribuições desta tese: duas heurísticas de pré-processamento do grafo do PGC, aplicado ao contexto de detecção de comunidades. O Capítulo 5 apresenta outra contribuição proposta nesta tese: uma formulação que dá origem a modelos mate-

máticos mais compactos para o PGC no contexto de agrupamento de dados qualitativos. No Capítulo 6 será apresentada a última contribuição proposta nesta tese: uma formulação que dá origem a modelos matemáticos mais compactos para o PGC aplicado ao problema de Edição de *Clusters*. Por fim, o Capítulo 7 apresenta as principais conclusões e os trabalhos futuros que podem ser realizados a partir das técnicas propostas nesta tese.

# 2

# CONCEITOS BÁSICOS

Este capítulo apresenta os principais conceitos da teoria de grafos utilizados nesta tese. O objetivo é estabelecer a terminologia e a notação matemática que foram adotadas. Os conceitos que foram apresentados neste capítulo podem ser encontrados em [26]. Conceitos específicos, utilizados no contexto de alguma das técnicas propostas nesta tese, serão introduzidos em seus respectivos capítulos.

## 2.1 DEFINIÇÃO DE UM GRAFO

Um *grafo* é definido como $G = (V(G), E(G))$, em que $V(G)$ representa o conjunto de *vértices* e $E(G)$ o conjunto de *arestas* de $G$. O número de vértices de $G$ é representado por $n = |V(G)|$, e o de arestas por $m = |E(G)|$.

As técnicas apresentadas neste trabalho consideram todos os grafos como *não direcionados*, ou seja, as arestas não possuem orientação de sentido. Uma aresta é representada como um par não ordenado $\{i, j\}$, sendo $i$ e $j$ dois vértices não necessariamente distintos de $V$, definidos como *incidentes* a $\{i, j\}$. Os vértices $i$ e $j$ são denominados como *pontas* da aresta. Quando $i$ e $j$ coincidem a aresta é denominada como *laço*. Dois vértices incidentes a uma aresta em comum são denominados *adjacentes*.

## 2.2 REPRESENTAÇÃO DE UM GRAFO

Uma *matriz de adjacências* $A = (a_{ij})_{nxn}$ pode ser utilizada para representar um grafo $G$, em que $a_{ij} = 1$ se $\{i, j\} \in E$ e $a_{ij} = 0$ caso contrário. Os laços são contabilizados como duas arestas.

Os grafos que possuem um valor de *peso* para suas arestas é denominado *ponderado*. Esses grafos são representados por uma *matriz de pesos* $W = (w_{ij})_{nxn}$, em que $w_{ij}$ representa o peso da aresta incidente aos vértices $i$ e $j$.

## 2.3 GRAU DE UM VÉRTICE E VIZINHANÇA

O conjunto de *vizinhos* de um vértice $i$, denotado como $N_i$, é composto por todos os vértices adjacentes a ele em $V(G)$.

O *grau* de um vértice $i$ representa o números de arestas de $E(G)$ incidentes ao vértice $i$. Esse valor é calculado a partir da matriz de adjacências $A$, sendo definido como $d_i = \sum_{j=1}^{n} a_{ij}$.

A *força* de um vértice $i$ representa a soma do peso das arestas de $E(G)$ incidentes ao vértice $i$. Esse valor é calculado a partir da matriz de pesos $W$, sendo definida como $s_i = \sum_{j=1}^{n} w_{ij}$.

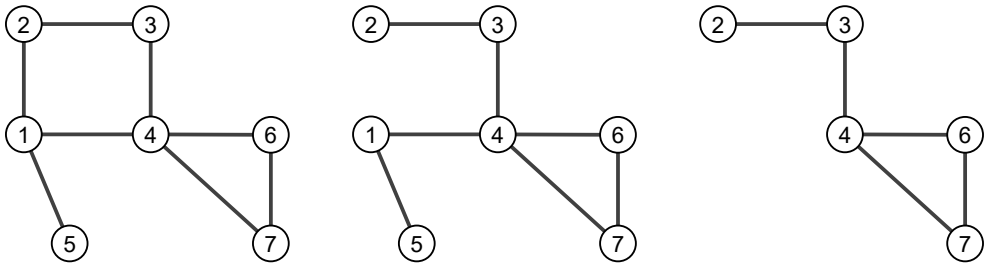
## 2.4 SUBGRAFOS

A Figura 1 será utilizada para ilustrar visualmente os conceitos apresentados nesta seção.

Um *subgrafo* de $G$, será escrito como $H \subseteq G$. Um subgrafo *gerador* de $G$ é aquele que contém todo o conjunto de vértices de $G$. A Figura 1b apresenta um exemplo de um subgrafo gerador criado a partir do grafo apresentado na Figura 1a.

Um *subgrafo induzido* de $G$ é aquele formado por um subconjunto dos vértices de $G$ e de todas as arestas de $G$ que conectam os pares desse subconjunto de vértices. A Figura 1c apresenta um exemplo de um subgrafo induzido criado a partir do grafo apresentado na Figura 1a.

Um *subgrafo maximal* de $G$ é aquele que não está estritamente contido em outro subgrafo.




(a) Grafo $G$. (b) Subgrafo gerador de $G$. (c) Subgrafo induzido de $G$

Figura 1: Exemplos de subgrafos.

## 2.5 CONECTIVIDADE

A Figura 2 será utilizada para ilustrar visualmente os conceitos apresentados nesta seção.

Uma *partição* de $V$ é uma coleção de subconjuntos não-vazios $\mathcal{P} = \{C_1, \ldots, C_n\}$, tal que a união de todos os membros de $\mathcal{P}$ é igual ao conjunto $V$. Esses subconjuntos são geralmente denominados como *grupos* ou *clusters*.

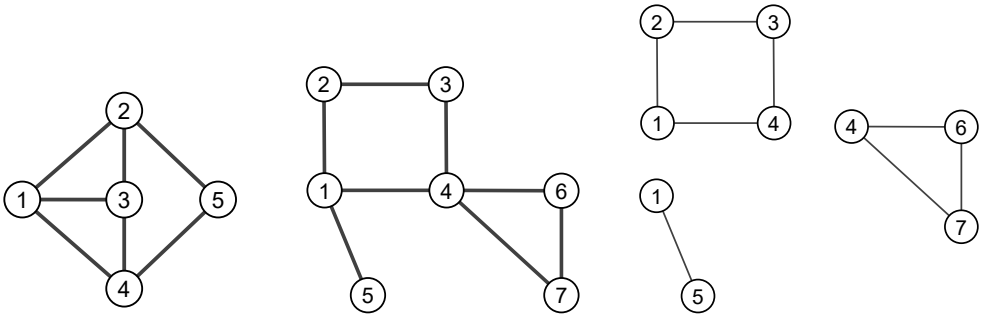
Um grafo é *conexo* se, para toda a partição do seu conjunto de vértices em dois conjuntos não-vazios $X$ e $Y$, existe uma aresta que possua uma de suas pontas em $X$ e outra em $Y$; caso contrário o grafo é *desconexo*.

Os *componentes conexos* de um grafo são os subgrafos conexos maximais deste grafo.

Um *ponto de articulação* de um grafo conexo é o vértice cuja remoção resulta em um grafo desconexo. Os vértices 1 e 4 do grafo apresentado na Figura 2b são pontos de articulação.

Um grafo *biconexo* é aquele que permanece conexo mesmo que qualquer um de seus vértices (e arestas incidentes) seja removido. Ou seja, ele não possui pontos de articulação. A Figura 2a representa um exemplo de grafo biconexo.

Os *componentes biconexos* ou *bicomponentes* de um grafo são os subgrafos biconexos maximais deste grafo. A Figura 2c apresenta os bicomponentes do grafo apresentado na Figura 2b.




(a) Grafo biconexo. (b) Grafo não biconexo. (c) Componentes biconexos.

Figura 2: Exemplo de um grafo biconexo, não biconexo e de componentes biconexos.

## 2.6 GRAFO COMPLETO E CLIQUES

Um grafo é *completo* quando todos os seus pares de vértices são adjacentes. A notação $K_n$ representará um grafo completo composto por $n$ vértices. A Figura 3 ilustra um grafo completo composto por 4 vértices.

Um grafo pode conter subgrafos completos denominados *cliques*. Na figura 3, um exemplo de subgrafo completo seria o clique formado pelos vértices 1, 2 e 3.

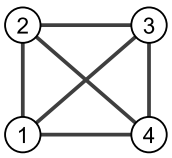


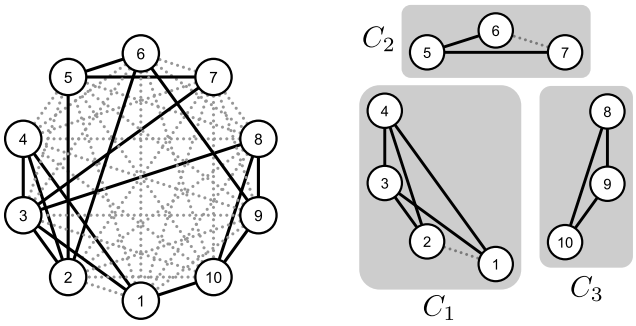

Figura 3: Exemplo de um grafo completo.

# 3

# PARTICIONAMENTO DE GRAFO EM CLIQUES

No problema denominado *Particionamento de Grafo em Cliques* (PGC) [42], o objetivo é particionar os vértices de um grafo completo ponderado $K_n = (V(K_n), E(K_n))$ em um subconjunto disjunto de vértices (grupos), tal que a soma dos pesos das arestas dentro dos grupos seja máxima. O problema recebe essa denominação pois os grupos obtidos são subgrafos completos (cliques). Grotschel e Wakabayashi [42] introduziram esse problema em 1989 e apresentaram uma formulação de Programação Linear Inteira (PLI) para solucioná-lo.

A Figura 4 apresenta um representação gráfica desse problema considerando um conjunto de dados formado por 10 entidades (Figura 4a). Cada vértice do grafo completo representa uma entidade que se deseja agrupar, e todas as entidades estão ligadas entre si através de arestas com peso definido de acordo com o tipo de aplicação. Nessa figura, as arestas que possuem valores de peso positivo são representadas por uma linha contínua, enquanto que as arestas que possuem valores de peso negativo são representadas por uma linha pontilhada. Na Figura 4b está representada uma solução possível para essa instância do problema.




(a) Grafo $K_{10}$. (b) Solução.

Figura 4: Representação gráfica do grafo completo de entrada do PGC e de uma solução possível para essa instância.

Geralmente, o peso das arestas do grafo $K_n$ representará uma estimativa da similaridade entre os vértices considerados, podendo ser um valor real ou inteiro. Neste trabalho, serão estudados problemas que definem o peso das arestas como valores reais (Capítulo 4) e inteiros (Capítulo 5 e 6).

A primeira etapa necessária para modelar um problema como PGC é criar o grafo completo $K_n$ a partir de um conjunto de dados formado por $n$ entidades. Quando o conjunto de dados está estruturado em formato tabular, em que cada linha representa uma entidade e as colunas representam seus atributos,

o grafo $K_n$ é criado diretamente. Uma entidade será representada por um vértice que estará ligado aos demais vértices do grafo através de arestas. O peso de uma aresta será definido através do cálculo da similaridade entre os atributos dos vértices incidentes a ela. O problema apresentado no Capítulo 5, por exemplo, é modelado dessa forma.

Em alguns casos, o conjunto de dados já está estruturado como um grafo $G$. Nesses casos, a adaptação para o PGC é feita da seguinte forma: cada vértice de $G$ será um vértice de $K_n$ e os pesos das arestas de $K_n$ podem ser definidos levando-se em conta as ligações existentes (ou não) entre os vértices de $G$. Os problemas apresentados no Capítulos 4 e 6, por exemplo, são modelados dessa forma.

A definição do peso das arestas do grafo $K_n$ é fundamental para a solução do PGC e depende do tipo de aplicação. Uma definição inadequada pode criar instâncias triviais do problema. Um exemplo de instância trivial ocorre, por exemplo, quando todas as arestas de $K_n$ possuem peso positivo. Nesse caso, de acordo com o objetivo do problema, a melhor solução será criar um único grupo composto por todos os vértices do grafo. Outro exemplo de solução trivial ocorre quando todas arestas de $K_n$ possuem peso negativo, neste caso a melhor solução é manter todos os vértices do grafo separados. Grotschel e Wakabayashi [42] provaram que quando o grafo $K_n$ possui arestas com peso positivo e negativo o problema é considerado NP-difícil.

Definido o grafo $K_n$, pode-se solucionar o problema de particionamento, de forma exata, através da formulação PLI proposta por Grotschel e Wakabayashi [42]:

$$(\mathrm{PGC}) \quad \mathrm{Maximizar} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_{ij}$$

$$\mathrm{sujeito\ a}$$

$$x_{ij} + x_{jk} - x_{ik} \leqslant 1 \quad , \ i < j < k \qquad (1)$$

$$x_{ij} - x_{jk} + x_{ik} \leqslant 1 \quad , \ i < j < k \qquad (2)$$

$$-x_{ij} + x_{jk} + x_{ik} \leqslant 1 \quad , \ i < j < k \qquad (3)$$

$$x_{ij} \in \{0, 1\} \quad , \ i < j$$

As variáveis de decisão $x_{ij}$ assumem o valor 1 se as entidades $i$ e $j$ estão no mesmo grupo, e 0 caso contrário. O coeficiente $w_{ij}$ representa o peso da aresta $\{i, j\}$ no grafo $K_n$. As restrições de transitividade (1-3) garantem que, para todo triângulo do grafo $K_n$, se duas arestas pertencem ao mesmo grupo então todo o triângulo pertence ao grupo.

A principal vantagem dessa formulação PLI é que a quantidade de grupos não precisa ser definida previamente pelo usuário, pois ela é obtida diretamente através da otimização da função objetivo do problema. A principal desvantagem é o grande número de variáveis ($O(n^2)$) e restrições de transitividade ($O(n^3)$).

O tamanho dos modelos matemáticos criados a partir dessa formulação impõe uma limitação ao seu uso. É necessário um grande esforço computacional (tempo e memória) para solucioná-los de forma exata. Apesar disso, Grotschel e Wakabayashi [42] demonstraram empiricamente, ao aplicarem uma técnica de *Planos de Corte* [87], que existe um número grande de restrições de transitividade redundantes. Os autores observaram que restrições redundantes não têm impacto direto na qualidade da solução do problema, mas têm impacto direto no desempenho do algoritmo que irá solucionar o problema de forma exata.

Trabalhos mais recentes da literatura indicam duas formas para tentar amenizar o problema dessa formulação:

1. Direta: identificar restrições de transitividade redundantes e criar uma formulação que evite que essas restrições sejam inseridas no modelo matemático.

2. Indireta: reduzir o tamanho do grafo $\mathrm{K_n}$, utilizado como entrada para o problema, através de uma técnica de pré-processamento.

Os trabalhos de Dinh e Thai [27] e Miyauchi e Sukegawa [67] são exemplos de como resolver o problema presente na formulação original de forma direta, enquanto que o trabalho de Arenas et al. [9] é uma importante contribuição para a forma indireta.

O trabalho de Dinh e Thai [27] foi pioneiro em tentar individualizar quais restrições de transitividade do PGC são redundantes, e criar uma nova formulação que evita a inclusão dessas restrições. Essa nova formulação cria modelos matemáticos que são solucionados de forma exata em menor tempo e consumindo menos memória que os modelos criados com a formulação original proposta por Grotschel e Wakabayashi [42]. Posteriormente, Miyauchi e Sukegawa [67] estenderam o trabalho de Dinh e Thai [27], criando uma formulação capaz de evitar um conjunto ainda maior de restrições redundantes, criando um modelo matemático mais compacto que demanda menos esforço em termos de tempo computacional e memória para ser solucionado de forma exata.

A formulação de Miyauchi e Sukegawa [67] é considerada, atualmente, a formulação capaz de criar os modelos matemáticos mais compactos para o PGC. Os autores ressaltaram, entretanto, que ainda existe um grande número de restrições de transitividade redundantes que sua formulação não foi capaz de evitar. Esse fato foi explorado ao longo dessa tese para propor formulações mais compactas para o PGC.

Arenas et al. [9] propuseram uma técnica capaz de reduzir o tamanho do grafo $\mathrm{K_n}$ sem prejudicar a estrutura de grupos presente no grafo original. O trabalho de Dinh e Thai [27] foi, novamente, o primeiro a aplicar essa técnica no contexto do método exato. A redução beneficia diretamente a solução do PGC através da formulação PLI, pois cria um modelo matemático com um número

menor de variáveis e restrições. Nesta tese, o trabalho de Arenas et al. [9] foi utilizado como base para a criação de duas técnicas de pré-processamento que reduzem ainda mais o tamanho do grafo $K_n$. Mais detalhes sobre a técnica de Arenas et al. [9] serão apresentados mais à frente no Capítulo 4.

Nas seções que seguem serão apresentados problemas de agrupamento de dados que foram resolvidos através do PGC, as técnicas que foram empregadas para resolver esse problema e detalhes sobre a formulação proposta por Miyauchi e Sukegawa [67].

## 3.1 APLICAÇÕES DO PGC E TRABALHOS RELACIONADOS

O PGC foi aplicado com sucesso em problemas como o agrupamento de dados qualitativos [42, 24, 31, 85, 19, 73, 90], agrupamento de documentos [10], agrupamento de máquinas e peças na área de tecnologia de grupo [84, 72], agendamento de aviões e portões em aeroportos [29, 30], agrupamento de proteínas por similaridade [57, 16, 17, 47, 12], detecção de comunidades [1, 5, 27, 28, 66], segmentação de consumidores no contexto de *marketing* [8], segmentação de imagens [56, 52, 53, 6, 7], identificação de registros duplicados em diferentes bases de dados [82], consenso entre partições [39, 3, 44], entre outros.

Devido ao elevado custo computacional para se resolver a formulação PLI original do PGC, outros algoritmos exatos foram utilizados na literatura, tais como: *Planos de Corte* [42, 72, 50, 17, 7, 52], *Branch-and-Bound* [75, 31], *Geração de Colunas* [47, 5], entre outros. Este trabalho não irá abordá-los, pois não foram utilizados nesta tese. Mais detalhes sobre esses algoritmos podem ser encontrados em Wolsey [87]. Esses algoritmos, entretanto, podem se beneficiar das técnicas apresentadas nesta tese. O Capítulo de conclusão desta tese (Capítulo 7) apresenta algumas sugestões de como utilizar esses algoritmos conjuntamente.

Como o PGC é NP-difícil, heurísticas também foram exploradas para solucioná-lo. Pode-se destacar as técnicas: metaheurísticas [24, 85, 19, 73, 90, 84], heurísticas gulosas [8, 39], heurísticas baseadas em programação linear [1, 56, 53, 3, 44], heurística híbrida [82], e técnicas que fazem uma redução prévia do grafo do problema e resolvem a formulação PLI de forma aproximada ou exata [6, 27, 28].

A Figura 5 apresenta em ordem cronológica um grafo de citações que relaciona os trabalhos referenciados acima. Os vértices representados como círculos correspondem a métodos exatos, enquanto que as heurísticas são representadas por quadrados. O estudo de cada um dos artigos ao longo do desenvolvimento desta tese ajudou a identificar o problema principal do PGC: a redundância presente na formulação PLI do PGC proposta por Grotschel e Wakabayashi [42].

Os vértices destacados em azul, na Figura 5, representam os trabalhos relacionados especificamente às técnicas que procuram melhorar o desempenho

dos modelos matemáticos criados a partir da formulação original PLI do PGC proposta por Grotschel e Wakabayashi [42]. Entre eles, estão em destaque os trabalhos de Arenas et al. [9], Dinh e Thai [27] e Miyauchi e Sukegawa [67].

## 3.2 FORMULAÇÕES PLI QUE CRIAM MODELOS MATEMÁTICOS MAIS COMPACTOS PARA O PGC

Dinh e Thai [27] propuseram uma formulação PLI capaz de evitar que restrições definidas como redundantes sejam inseridas. Segundo os autores, essa abordagem é mais vantajosa que a técnica de planos de corte proposta em [42], pois a exclusão de maior parte da redundância é feita diretamente na formulação e não envolve a resolução iterativa de relaxações do problema original.

Recentemente, Miyauchi e Sukegawa [67] generalizaram os resultados de Dinh e Thai [27] e propuseram uma formulação PLI mais compacta. Os autores analisaram os grupos de uma partição ótima (Figura 6a) e definiram uma condição que o peso das arestas dentro do grupo devem respeitar para que a estrutura do grupo seja preservada. Os autores provaram que para qualquer subdivisão de um grupo em dois subgrupos $A$ e $B$ (Figura 6b), deve existir pelo menos uma aresta de peso não negativo no conjunto de arestas entre os subgrupos ($E_{AB}$). Caso contrário, esse grupo poderia ser subdividido, melhorando a solução ótima, o que seria uma contradição.

Para garantir que exista pelo menos uma aresta com peso não negativo no conjunto $E_{AB}$ foram inseridas cláusulas condicionais nas restrições de transitividade (1-3) da formulação $PGC$. A nova formulação pode ser escrita como:

$$(PGC_R)\ Maximizar \quad \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}x_{ij}$$

$$sujeito\ a$$

$$x_{ij} + x_{jk} - x_{ik} \leqslant 1 \quad , i < j < k,\ w_{ij} \geqslant 0 \vee w_{jk} \geqslant 0 \tag{14}$$

$$x_{ij} - x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k,\ w_{ij} \geqslant 0 \vee w_{ik} \geqslant 0 \tag{15}$$

$$-x_{ij} + x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k,\ w_{jk} \geqslant 0 \vee w_{ik} \geqslant 0 \tag{16}$$

$$x_{ij} \in \{0,1\} \quad , i < j$$

Cada restrição de transitividade corresponde a um triângulo do clique $K_n$. Restrições que envolvem triângulos que possuem pesos de arestas que não respeitem as cláusulas condicionais das restrições (4-6) serão considerados redundantes e não serão inseridas no modelo matemático.

A explicação para a cláusula condicional adicionada à primeira restrição da formulação é apresentada na Figura 7. A Figura 7a mostra o triângulo correspondente a restrição 4, enquanto que a Figura 7b apresenta a cláusula condicional que define que pelo menos uma aresta com peso não negativo deve existir entre os conjuntos $A = \{j\}$ e $B = \{i, k\}$. O mesmo raciocínio serve

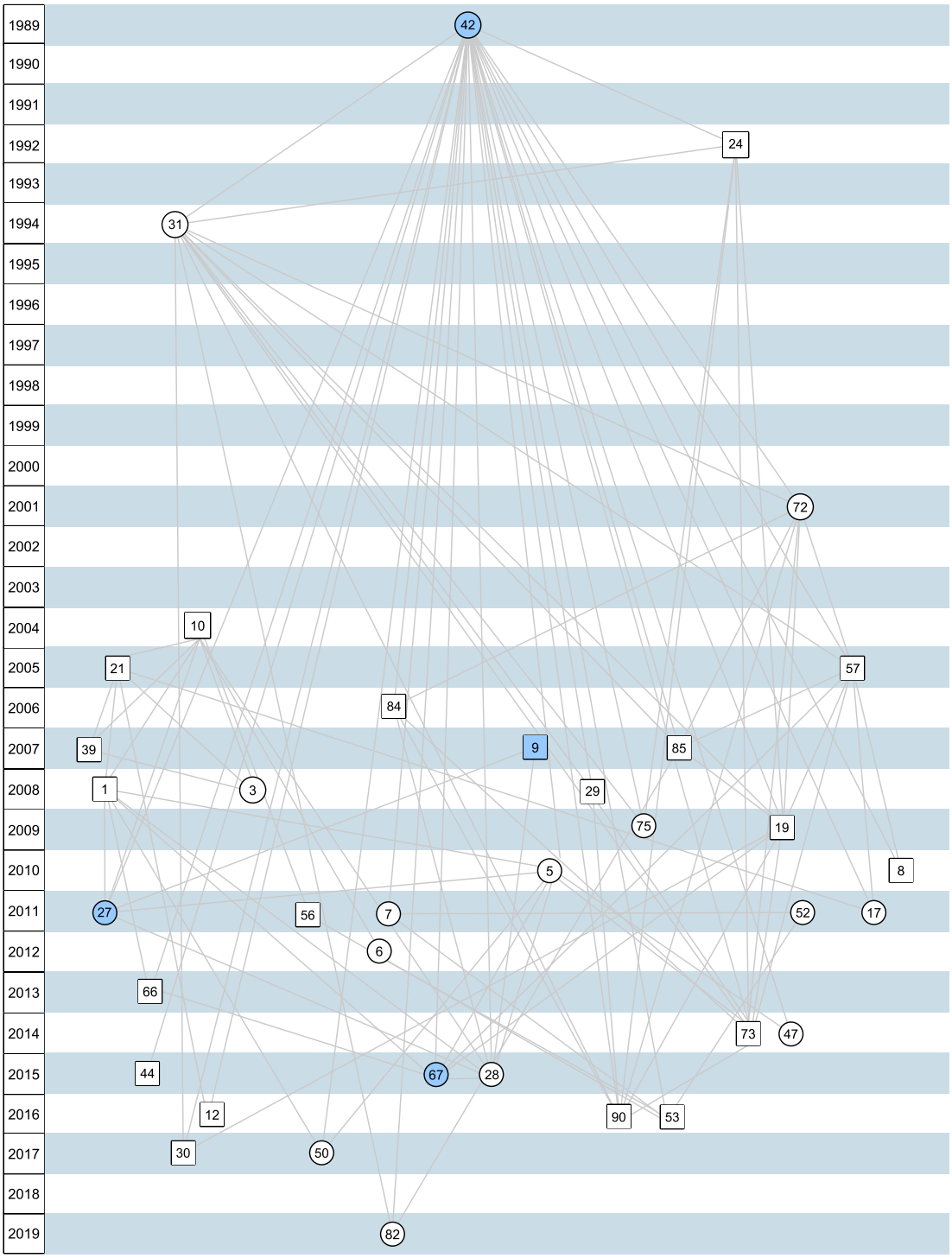
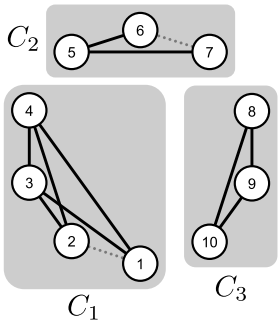
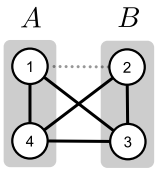



Figura 5: Grafo de citações criado a partir das referências relacionadas ao PGC, organizado em ordem cronológica. Quadrados representam heurísticas e círculos representam métodos exatos.



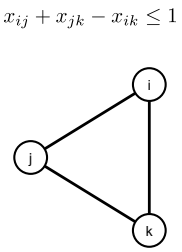

(a) Partição Ótima.



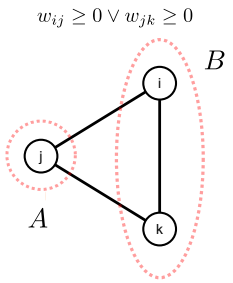

(b) Subdivisão de $C_1$.

Figura 6: Análise de um grupo de uma partição ótima.

para as demais restrições de transitividade da formulação $\mathrm{PGC_R}$, uma restrição será incluída na formulação apenas se sua respectiva condição for respeitada, caso contrário a restrição será considerada redundante.




(a) Restrição.




(b) Cláusula condicional.

Figura 7: Triângulo correspondente à restrição de transitividade 4 e respectiva cláusula condicional.

A estratégia proposta por Miyauchi e Sukegawa [67] é capaz de reduzir o número de restrições de transitividade redundantes, criando um modelo matemático que demanda menor esforço computacional para ser solucionado de forma exata. Existe, entretanto, um grande número de restrições de transitividade redundantes que essa formulação não foi capaz de evitar. Esse fato foi explorado nos próximos capítulos para propor formulações mais compactas para o PGC.

# 4

# PGC E DETECÇÃO DE COMUNIDADES

O termo *Detecção de Comunidades* é frequentemente empregado como sinônimo para agrupamento de dados em grafos em diversos campos da ciência, como física, química, biologia, sociologia e computação [81, 4, 68]. Neste capítulo, uma comunidade será definida como um grupo de vértices densamente conectados entre si [70].

A detecção de comunidades é um problema ativamente estudado. Várias técnicas foram propostas na literatura, de heurísticas a métodos exatos [33, 34]. Agarwal e Kempe [1] foram os primeiros a utilizarem a formulação PGC apresentada no Capítulo 3 no contexto de detecção de comunidades. A formulação original do PGC, entretanto, cria modelos matemáticos com um número excessivo de variáveis e restrições.

Uma forma de tentar mitigar o problema do tamanho dos modelos matemáticos criados a partir da formulação PGC é tentar reduzir o grafo do problema através de uma técnica de pré-processamento. Esse capítulo propõe duas novas técnicas de pré-processamento (*LCCFilter* e *BCCFilter*), que estendem a técnica proposta por Arenas et al. [9], obtendo resultados superiores em relação a capacidade de redução do grafo do problema.

Os resultados experimentais mostram que os grafos reduzidos pelas duas técnicas propostas nesta tese resultam em modelos matemáticos mais compactos (menos variáveis e restrições), e que demandam um menor esforço computacional (tempo e memória) para serem solucionados de forma exata.

## 4.1 DETECÇÃO DE COMUNIDADES POR MAXIMIZAÇÃO DA MODULARIDADE

Um grande número de técnicas baseiam-se na otimização de uma função de qualidade. A modularidade, proposta por Newman e Girvan [70], é a função de qualidade mais popularmente utilizada para estimar a qualidade de uma partição obtida por uma técnica de detecção de comunidades.

Para calcular a modularidade, assume-se que os vértices de um grafo ponderado $G$ foram particionados por uma técnica de detecção de comunidades. O resultado desse processo é uma partição $\mathcal{P} = \{C_1, C_2, \ldots, C_p\}$ composta por $p$ subconjuntos disjuntos de vértices definidos como comunidades. A modularidade dessa partição, no contexto de um grafo ponderado, é definida como:

$$Q(\mathcal{P}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left( \frac{w_{ij}}{2s} - \frac{s_i s_j}{4s^2} \right) \delta_{ij} \tag{7}$$

em que $\delta_{ij} = 1$ se os vértices $i$ e $j$ estão na mesma comunidade, e $0$ caso contrário. Nesta equação, $w_{ij}$ representa o peso da aresta $\{i, j\}$. Os valores de $s_i$ e $s_j$ representam a força dos vértices $i$ e $j$, enquanto que $s$ representa a soma do peso de todas as arestas do grafo. O valor de $Q$ varia no intervalo $[-\frac{1}{2}, 1]$. Consequentemente, quanto maior o valor da modularidade, melhor será a qualidade da partição [70, 69].

## 4.2 MAXIMIZAÇÃO DE MODULARIDADE VIA PLI

O problema de detecção de comunidades via maximização de modularidade pode ser resolvido de maneira exata através da formulação PLI do PGC proposta por Grotschel e Wakabayashi [42], adaptando-se a função objetivo do problema. A função objetivo é derivada da fórmula de modularidade apresentada na Equação (7). Variáveis binárias $x_{ij}$ são introduzidas, assumindo um valor igual a 1 se $i$ e $j$ pertencem à mesma comunidade e $0$, caso contrário.

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{w_{ij}}{2s} - \frac{s_i s_j}{4s^2}\right) x_{ij}$$

A expressão entre parênteses é um elemento da matriz de modularidade $M = (m_{ij})_{nxn}$. Consequentemente, a função objetivo pode ser simplificada para

$$\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}\, x_{ij}$$

Variáveis $x_{ii}$ podem ser substituídas por uma constante na função objetivo, uma vez que $x_{ii} = 1$ para todos os vértices:

$$C = \sum_{i=1}^{n} m_{ii}$$

As variáveis $x_{ij}$ em que $i > j$ podem ser removidas da formulação, pois $x_{ij} = x_{ji}$ para todo $i$ e $j$. Elas são removidas ao se duplicar o valor da somatória que considera tais variáveis na função objetivo, resultando na seguinte formulação PLI:

$$(PMM)\ Maximizar \quad 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} m_{ij}x_{ij} + C$$

$$sujeito\ a$$

$$x_{ij} + x_{jk} - x_{ik} \leqslant 1 \quad , i < j < k \tag{8}$$

$$x_{ij} - x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k \tag{9}$$

$$-x_{ij} + x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k \tag{10}$$

$$x_{ij} \in \{0, 1\} \quad , i < j$$

Pode-se observar que a formulação obtida é exatamente a formulação PGC apresentada no Capítulo 3. As mesmas restrições de transitividade (8-10) são utilizadas, apenas a função objetivo foi adaptada ao contexto de maximização da modularidade. Essa formulação, portanto, possui as mesmas deficiências apontadas previamente.

Formulações que criam modelos matemáticos mais compactos foram propostas no contexto de maximização da modularidade [27, 67] para tentar amenizar tais deficiências. Neste capítulo, utilizaremos a formulação $\mathrm{PGC_R}$ proposta por Miyauchi e Sukegawa [67], também apresentada no Capítulo 3. Ela acrescenta cláusulas condicionais às restrições de transitividade da formulação $\mathrm{PMM}$, dando origem a seguinte formulação:

$$
\begin{aligned}
(\mathrm{PMM_R}) \quad \textit{Maximizar} \quad & 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} m_{ij}x_{ij} + C \\
\textit{sujeito a} \quad & \\
& x_{ij} + x_{jk} - x_{ik} \leqslant 1 \quad , i < j < k \quad , m_{ij} \geqslant 0 \vee m_{jk} \geqslant 0 \\
& x_{ij} - x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k \quad , m_{ij} \geqslant 0 \vee m_{ik} \geqslant 0 \\
& -x_{ij} + x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k \quad , m_{jk} \geqslant 0 \vee m_{ik} \geqslant 0 \\
& x_{ij} \in \{0,1\} \quad , i < j
\end{aligned}
$$

## 4.3 TÉCNICAS DE PRÉ-PROCESSAMENTO

Uma maneira de se reduzir o custo computacional dos modelos matemáticos criados a partir da formulação $\mathrm{PMM_R}$ é reduzir o tamanho do grafo do problema, através de técnicas de pré-processamento. Pesquisas recentes propuseram técnicas de pré-processamento que podem manter determinadas propriedades estruturais do grafo, como a conectividade, cortes, fluxo, propriedades espectrais e modularidade [18, 38, 40, 55, 88, 80, 9].

O trabalho de Arenas et al. [9] é particularmente interessante no contexto da maximização da modularidade. O restante desta seção será dedicado à análise das principais contribuições desse trabalho, pois as duas técnicas de pré-processamento propostas neste capítulo são derivações da técnica proposta em [9].

Inicialmente, os autores em [9] demonstraram que é possível criar uma representação reduzida de um grafo a partir de uma partição sem que a modularidade seja prejudicada. A Figura 8 será utilizada para exemplificar esse conceito. A partir de uma partição $\mathcal{P} = \{C_1, C_2, C_3, C_4\}$ de $G$ (Figura 8a), é criada a representação reduzida $G'$ apresentada na Figura 8b. Nessa representação, os vértices participantes de uma mesma comunidade em uma partição da Figura 8a são contraídos em um único vértice, que armazenará no peso de seu laço a soma do peso das arestas dentro da comunidade. Arestas existentes entre os vértices dessa comunidade com as demais comunidades da partição da Figura 8a são contraídas em uma única aresta, que terá como peso a soma

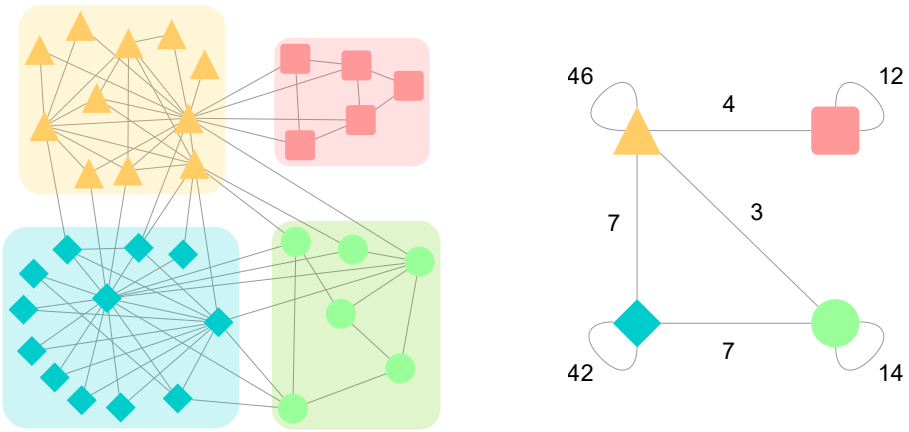
do peso das arestas contraídas. Os pesos das arestas de $G'$ foram definidos pelos autores através das regras que seguem.

No grafo $G'$, um vértice $k$ representa uma comunidade $C_k$ de $\mathcal{P}$. O peso de seu laço é definido como:

$$w_{kk} = 2\left(\sum_{i \in C_k} \sum_{j \in C_k, i \neq j} w_{ij}\right) + \sum_{i \in C_k} w_{ii} \tag{11}$$

Dados dois vértices $k$ e $l$ de $G'$, representantes das comunidades $C_k$ e $C_l$ da partição $\mathcal{P}$, o peso da aresta $\{k, l\}$ é definido:

$$w_{kl} = \sum_{i \in C_k} \sum_{j \in C_l} w_{ij} \tag{12}$$




(a) Partição ótima do grafo $G$.

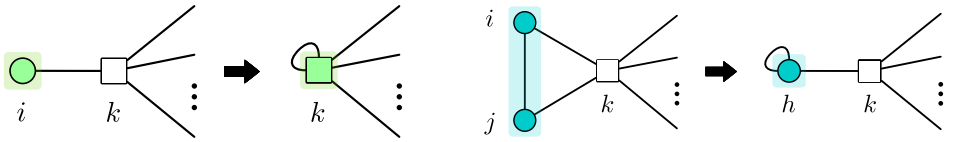
(b) Grafo reduzido $G'$.

Figura 8: Partição do grafo *Zachary's karate club* [89] e sua representação reduzida.

Os autores provaram que a modularidade da partição ótima do grafo $G$ é igual a modularidade da partição ótima correspondente no grafo reduzido $G'$. A partir desse resultado, os autores tentaram identificar conjuntos de vértices que sempre estarão juntos em uma partição ótima e pré-agrupá-los em uma representação reduzida. Foi provado que dois tipos de subgrafos periféricos podem ser pré-agrupados: cliques $K_1$ e $K_2$ (Figura 9).

Na Figura 9a, o vértice $i$ do subgrafo $K_1$ pode ser pré-agrupado ao vértice $k$, pois os autores provaram que o vértice $i$ nunca ficará isolado em uma partição ótima. O peso do laço do vértice $k$ é definido segundo a Equação (11), resultando em $w_{kk} = 2w_{ik}$.

Para o subgrafo $K_2$ (Figura 9b), os autores provaram que os vértices $i$ e $j$ podem ser pré-agrupados em uma comunidade representada pelo vértice $h$, com um laço de peso definido pela Equação (11), resultando em $w_{hh} = 2w_{ij}$. As arestas entre os vértices do subgrafo $K_2$ e o vértice $k$ são contraídas em uma única aresta $\{h, k\}$ com peso definido pela Equação (12), resultando em $w_{hk} = w_{ik} + w_{jk}$.




(a) Clique $K_1$. (b) Clique $K_2$.

Figura 9: Tipos de subgrafos periféricos pré-processados por Arenas et al. [9].

Ao reduzir o tamanho do grafo do problema, os autores obtiveram um ganho em desempenho computacional na heurística de detecção de comunidades empregada por eles durante os experimentos computacionais. Além disso, a qualidade das soluções obtidas, medida através da modularidade, não foi prejudicada.

Nas conclusões do trabalho [9], os autores conjecturaram que talvez existam outros tipos de subgrafos que podem ser pré-processados sem o prejuízo para a modularidade. Eles ressaltaram, entretanto, que provar que a modularidade será inalterada nesses casos talvez seja uma tarefa difícil.

Nas seções que seguem serão apresentados os detalhes das duas heurísticas propostas nesta tese (*LCCFilter* e *BCCFilter*). Elas são capazes de pré-processar subgrafos periféricos maiores que a técnica proposta por Arenas et al. [9]. A heurística *LCCFilter* é capaz de pré-processar cliques periféricos de qualquer tamanho, enquanto que a heurística *BCCFilter* é capaz de pré-processar qualquer tipo de subgrafo periférico.

No restante desse capítulo, um subgrafo será considerado periférico se for ligado ao componente principal do grafo apenas por um ponto de articulação. O termo *filtro* será utilizado como sinônimo para uma técnica de pré-processamento, e o verbo *filtrar* representará o processo de pré-processamento do grafo.

## 4.4 FILTRO LCCFILTER

Nesta seção, será apresentada uma técnica que filtra cliques periféricos de qualquer tamanho. A Figura 10 apresenta um grafo em que foram destacadas quatro cliques periféricos, cliques $K_1$ em verde, $K_2$ em azul e $K_5$ em amarelo. Pode-se observar pela figura que cliques $K_1$ são facilmente identificados pelo grau do vértice participante desse subgrafo (grau 1). No entanto, para identificar vértices participantes de cliques $K_s$ com $s \geqslant 2$, somente a informação do grau dos vértice não ajuda a diferenciar se o vértice participa de um subgrafo periférico ou não. Este trabalho, portanto, utilizou a medida *Local Clustering Coefficient* (LCC) [86] de um vértice para identificar os vértices participantes de cliques periféricas.

O LCC de um vértice quantifica o quão conectado estão os seus vizinhos. Essa medida assume o valor máximo $(1,0)$ quando todos os vizinhos de um

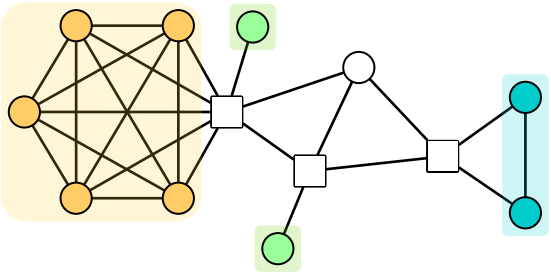


Figura 10: Exemplo de grafo com quatro cliques $K_s$ periféricos.

vértice estão conectados (clique), e o valor mínimo $(0,0)$ quando não há conexões entre os vizinhos. O LCC é definido como:

$$LCC(i) = \frac{\text{total de triângulos que o vértice } i \text{ participa}}{\text{número máximo de triângulos que o vértice } i \text{ poderia participar}} \tag{13}$$

Para o cálculo do numerador da Equação (13), pode-se utilizar qualquer algoritmo de contagem de triângulos disponível na literatura. O denominador de Equação 13 é definido como $\frac{d_i(d_i-1)}{2}$.

O algoritmo *Edge-Iterator* (Algoritmo 1) proposto por Schank e Wagner [77] foi modificado, neste trabalho, para calcular o LCC dos vértices de um grafo. Ele conta, para cada aresta $\{i,j\}$ de um grafo, os vizinhos que $i$ e $j$ tem em comum (triângulos). A complexidade assintótica desse algoritmo é $O(mn)$.

**Algoritmo 1:** *Edge-Iterator*

**entrada :** Um grafo ponderado $G = (V, E)$
**saída :** O vetor LCC contendo os valores de *Local Clustering Coefficient* de cada vértice

```
1  for i ← 1 to n do
2      LCC[i] ← 0.0
3  for i ← 1 to n do
4      for j ∈ N_i do
5          if i < j then
6              for k ∈ N_i ∩ N_j do
7                  if j < k then
8                      LCC[i] ← LCC[i] + 1
9                      LCC[j] ← LCC[j] + 1
10                     LCC[k] ← LCC[k] + 1
11 for i ← 1 to n do
12     if d_i ≠ 1 then
13         LCC[i] ← 2 * LCC[i]/(d_i * (d_i − 1))
14 return LCC
```

Utilizando o LCC, podemos observar, por exemplo, na Figura 11 que cliques com $s \geqslant 2$ são compostos por vértices com LCC igual a $1,0$. O único vizinho dos vértices desse subgrafo que não possui o valor de LCC igual a $1,0$ é o ponto de articulação. Essa propriedade foi utilizada como ponto de partida para criar o algoritmo de filtragem de cliques (Algoritmo 2).

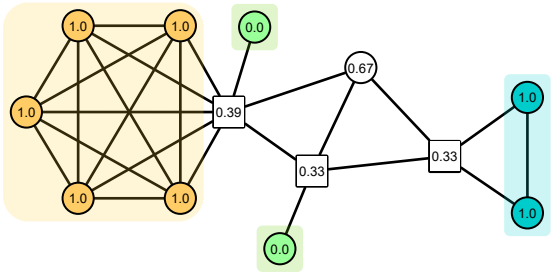



Figura 11: Identificando cliques $K_s$ através do LCC dos vértices.

O Algoritmo 2 apresenta a técnica *Local Clustering Coefficient Filter* (LCCFilter). Ela recebe o grafo do problema como entrada e produz uma representação reduzida $G'$. O algoritmo é composto de quatro etapas. A Figura 12 apresenta o comportamento de cada etapa do Algoritmo 2 em um grafo exemplo.

---

**Algoritmo 2:** *Local Clustering Coefficient Filter (LCCFilter).*

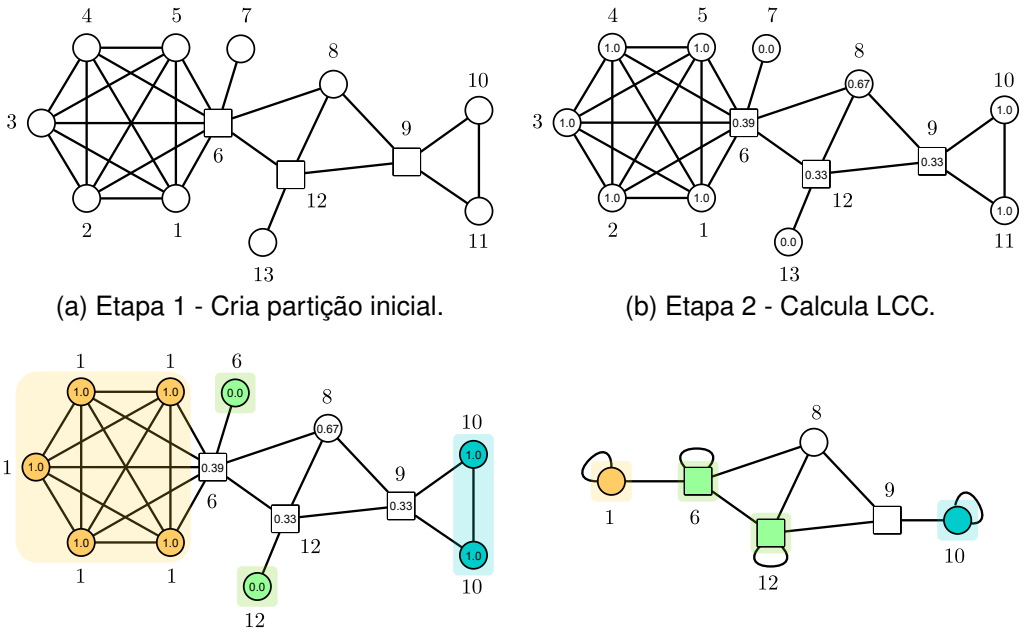
---

**entrada :** Um grafo ponderado $G = (V, E)$
**saída :** Um grafo ponderado $G' = (V', E')$

1. Criar uma partição inicial em que cada vértice participa de uma comunidade
2. Calcular o LCC de todos os vértices de $G$ através do Algoritmo 1
3. Para todo vértice em $G$:
   3a. Se o vértice tiver grau 1
      3aa. Atribuí-lo à mesma comunidade de seu vizinho
   3b. Se o vértice tiver LCC igual a 1,0 e todos os seus vizinhos, exceto um, tiverem LCC igual a 1,0
      3bb. Atribuir o vértice e todos os seus vizinhos com LCC igual 1,0 à uma mesma comunidade
4. Criar uma representação menor do grafo ($G'$) a partir das comunidades da partição obtida

---

A primeira etapa do algoritmo (Figura 12a) cria uma partição inicial em que cada vértice de $G$ participa de uma comunidade. Essa etapa é realizada pois os vértices que estiverem na mesma comunidade na etapa final do algoritmo serão filtrados. Na segunda etapa (Figura 12b), calcula-se o LCC de todos os vértices de $G$ através do Algoritmo *Edge-Iterator* 1. A terceira etapa (Figura 12c) testa se cada vértice de $G$ pertence a um clique $K_s$. Um vértice com grau 1 é participante de uma clique $K_1$, e será atribuído para a mesma comunidade




(a) Etapa 1 - Cria partição inicial.

(b) Etapa 2 - Calcula LCC.

(c) Etapa 3 - Identifica cliques $K_s$.

(d) Etapa 4 - Cria representação reduzida $G'$.

Figura 12: Exemplo ilustrando cada etapa do Algoritmo 2.

que seu vizinho. Um vértice com LCC igual a $1,0$ e que possua $|N_i| - 1$ vizinhos com valores de LCC igual a $1,0$ é participante de uma clique $K_s$. Esse vértice e todos os seus vizinhos com LCC igual a $1,0$ são atribuídos à mesma comunidade. Finalmente, na etapa 4, a estratégia de contração de vértices e arestas (Equação 11 e 12) discutida na Seção 4.3 é utilizada para criar uma representação reduzida $G'$ (Figure 12d). Os vértices que não foram filtrados, preservam suas propriedades estruturais em $G'$.

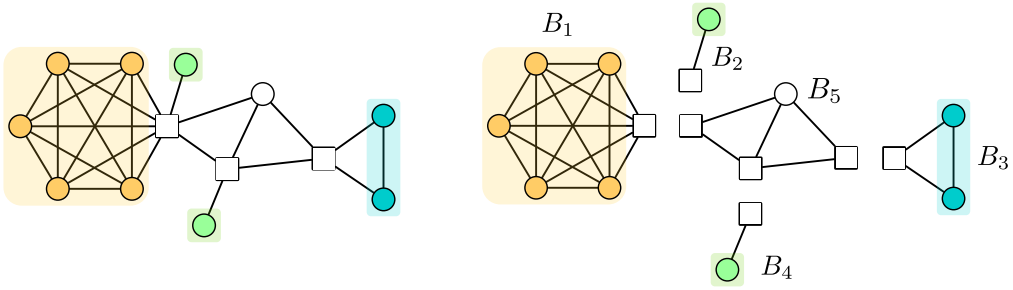
A complexidade assintótica do Algoritmo 2 é $O(mn)$ (Algoritmo 1). O algoritmo para o cálculo LCC, utilizado na etapa 2, é o fator dominante. As etapas restantes têm complexidade de tempo linear.

A representação reduzida do grafo obtida pelo algoritmo *LCCFilter* pode ser utilizada diretamente na formulação $PMM_R$ descrita na seção 4.2. A seção de resultados experimentais mostrará como que esta representação pode beneficiar o modelo exato. Na próxima seção será apresentada uma técnica capaz de filtrar outros tipos de subgrafos periféricos, além de cliques.

## 4.5 FILTRO BCCFILTER

O filtro *LCCFilter*, além de ser indicado somente para cliques periféricos, possui complexidade assintótica $O(mn)$. Nesta seção será apresentado um novo filtro denominado *BCCFilter*, cuja complexidade assintótica é $O(n + m)$ e consegue filtrar cliques e outros tipos de subgrafos periféricos.

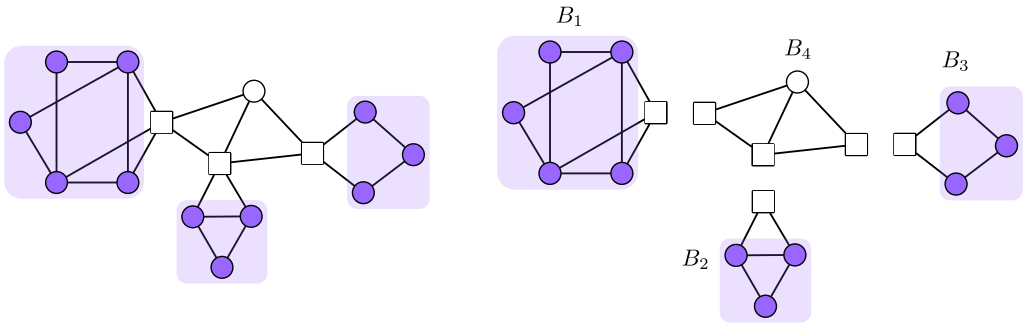
Para derivar um novo filtro, observou-se que os subgrafos filtrados por *LCCFilter* possuem uma característica em comum: são ligados ao componente principal do grafo por meio de um ponto de articulação. Se o grafo possui pontos de articulação, ele pode ser decomposto em bicomponentes. A Figura 13 apresenta a decomposição de um grafo $\mathrm{G}$ em bicomponentes. Pode-se observar que os bicomponentes das cliques periféricas ($\mathrm{B}_1 - \mathrm{B}_4$) se diferenciam do bicomponente $\mathrm{B}_5$ por apresentarem em seu conjunto de vértices todos os vértices do subgrafo periférico mais um ponto de articulação em $\mathrm{G}$.




(a) Grafo $\mathrm{G}$. (b) Bicomponentes de $\mathrm{G}$.

Figura 13: Identificando os bicomponentes do grafo $\mathrm{G}$. Vértices que são pontos de articulação em $\mathrm{G}$ são representados como um quadrado.

Além de cliques, existem outros tipos de subgrafos que podem se ligar a $\mathrm{G}$ por apenas um ponto de articulação. A Figura 14 apresenta uma decomposição em bicomponentes em que os bicomponentes $\mathrm{B}_1$, $\mathrm{B}_2$ e $\mathrm{B}_3$ são subgrafos periféricos.




(a) Grafo $\mathrm{G}$. (b) Bicomponentes de $\mathrm{G}$.

Figura 14: Identificação de outros tipos de subgrafos periféricos através da decomposição de $\mathrm{G}$ em bicomponentes. Vértices que são pontos de articulação em $\mathrm{G}$ são representados como um quadrado.

Pode-se observar, através das Figuras 13 e 14, que a decomposição do grafo em bicomponentes facilita a identificação de cliques e outros subgrafos periféricos. Essa característica, portanto, foi utilizada na proposição de um novo filtro, em que um grafo $\mathrm{G}$ é inicialmente decomposto em bicomponentes. Os bicomponentes representantes dos subgrafos periféricos são compostos

por todos os vértices do subgrafo e o ponto de articulação ao qual o subgrafo se liga em $G$. Para se encontrar os bicomponentes de um grafo, foi utilizado o algoritmo proposto por Hopcroft e Tarjan [46].

O Algoritmo 3 apresenta a técnica de pré-processamento *Biconnected Components Filter* (BCCFilter). Ela recebe o grafo do problema como entrada e produz uma representação reduzida $G'$. O algoritmo é composto de quatro etapas. A Figura 15 apresenta o comportamento de cada etapa do Algoritmo 3 em um grafo exemplo.

**Algoritmo 3:** *Biconnected Components Filter (BCCFilter).*

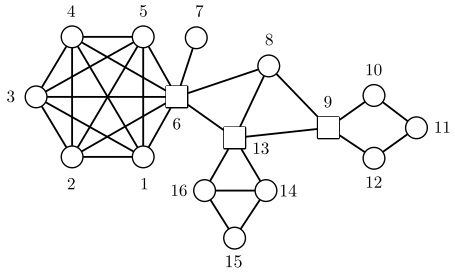
**entrada :** Um grafo ponderado $G = (V, E)$
**saída :** Um grafo ponderado $G' = (V', E')$

1. Criar uma partição inicial em que cada vértice participa de uma comunidade
2. Decompor o grafo $G$ em um conjunto de bicomponentes $BCC$
3. Para todo bicomponente $B \subseteq BCC$:
   3a. Checar se apenas um dos vértices em $B$ é um ponto de articulação em $G$
      3aa. Em caso afirmativo, atribua todos os vértices de $B$, exceto o ponto de articulação, à uma mesma comunidade
4. Criar uma representação menor do grafo ($G'$) a partir das comunidades da partição obtida

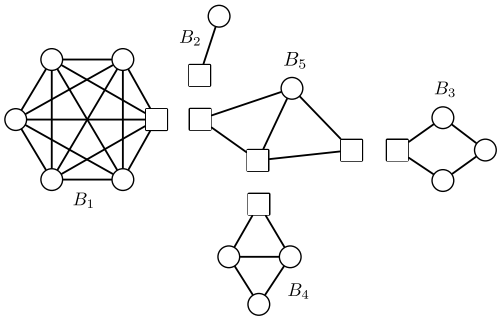
Na primeira etapa do algoritmo (Figura 15a), cria-se uma partição inicial em que cada vértice de $G$ participa de uma comunidade. Os vértices que estiverem na mesma comunidade na etapa final do algoritmo serão filtrados. Na segunda etapa (Figura 15b), o grafo é decomposto em bicomponentes utilizando o algoritmo desenvolvido por Hopcroft e Tarjan [46]. Na terceira etapa (Figura 15c) testa-se cada bicomponente obtida, verificando se ela corresponde a um subgrafo periférico. Os vértices participantes de um subgrafo periférico são atribuídos a uma mesma comunidade. Finalmente, na etapa 4, a estratégia de contração de vértices (Equação 11 e 12) discutida na Seção 4.3 é utilizada para criar uma representação reduzida $G'$ (Figure 15d). Os vértices que não foram filtrados, preservam suas propriedades estruturais em $G'$.

A complexidade assintótica para Algoritmo 3 é $O(m + n)$. O algoritmo de Hopcroft e Tarjan [46], utilizado na etapa 2, é o fator dominante. As etapas restantes têm complexidade de tempo linear.

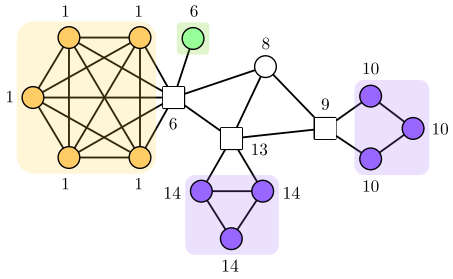
Assim como o *LCCFilter*, a representação reduzida obtida pelo algoritmo *BCCFilter* pode ser utilizada diretamente na formulação $PMM_R$ descrita na seção 4.2. A seção de resultados experimentais irá comparar o desempenho das duas técnicas.



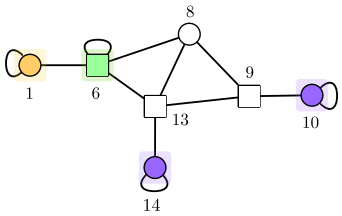

(a) Etapa 1 - Cria partição inicial.




(b) Etapa 2 - Decompõe o grafo em bicomponentes.




(c) Etapa 3 - Identifica bicomponentes que representam subgrafos periféricos.




(d) Etapa 4 - Cria representação reduzida G′.

Figura 15: Exemplo ilustrando cada etapa do Algoritmo 3.

## 4.6 RESULTADOS EXPERIMENTAIS

Nesta seção, serão apresentados os experimentos que foram realizados para verificar a validade dos filtros *LCCFilter* e *BCCFilter*. Inicialmente, ambos os filtros foram comparados em relação ao tempo computacional e capacidade de reduzir o tamanho do grafo do problema.

Em uma segunda etapa, foram criados modelos matemáticos baseados na formulação $\mathrm{PMM_R}$ a partir do grafo original e dos grafos reduzidos pelas técnicas de pré-processamento. Esses modelos foram solucionados de forma exata e o desempenho da técnica empregada em sua solução foi avaliado em termos de qualidade da solução obtida (modularidade), tempo computacional, tamanho do modelo matemático (variáveis e restrições) e consumo de memória.

Os experimentos e algoritmos foram codificados em C++ e executados em um computador com a seguinte configuração: Intel Core i7-6770HQ (3,5GHz) com 32 GB RAM rodando Windows 10 64-Bit. O software comercial *IBM ILOG CPLEX* [48] foi utilizado para resolver os modelos matemáticos.

A Tabela 1 apresenta os conjuntos de dados utilizados nos experimentos. Eles podem ser encontrados no repositório de dados *Network Repository* [76]. Cada conjunto de dados recebeu um identificador único (primeira coluna) e

Tabela 1: Conjuntos de dados utilizados nos experimentos.

| ID | Nome | n | m |
|---|---|---|---|
| 1 | soc-karate | 34 | 78 |
| 2 | lesmis | 77 | 254 |
| 3 | GD00-A | 83 | 125 |
| 4 | ca-sandi-auths | 86 | 124 |
| 5 | rt-retweet | 96 | 177 |
| 6 | netscience | 379 | 914 |
| 7 | bio-DM-LC | 483 | 997 |
| 8 | power-494-bus | 494 | 586 |
| 9 | bio-diseasome | 516 | 1 188 |
| 10 | bio-grid-mouse | 791 | 1 098 |
| 11 | ca-CSphd | 1 025 | 1 043 |

um nome (segunda coluna). As demais colunas apresentam o número de vértices $(\mathrm{n})$ e arestas $(\mathrm{m})$ do grafo que representa o problema. Apenas o maior componente conexo do grafo foi utilizado nos experimentos.

### 4.6.1 *Desempenho dos filtros LCCFilter e BCCFilter*

Nesta subseção, será apresentado o desempenho dos filtros *LCCFilter* e *BCCFilter* em relação ao tempo computacional e capacidade de reduzir o tamanho do grafo do problema.

A Tabela 2 apresenta o tempo computacional (segundos) e os tipos de subgrafos filtrados por cada técnica. O símbolo $\mathrm{K}_s$ é utilizado para representar cliques, enquanto que $\mathrm{H}_s$ é utilizado para outros tipos de subgrafos periféricos. O índice $s$ representa o número de vértices participantes do subgrafo, e o valor entre parênteses representa a quantidade de subgrafos periféricos desse tipo que foram filtrados pela técnica.

Pode-se observar, a partir da Tabela 2, que o filtro *LCCFilter* é capaz de filtrar cliques com $\mathrm{s} \geqslant 3$ para os conjuntos de dados 4, 6, 9. O filtro *BCCFilter*, além de filtrar cliques, conseguiu encontrar outros tipos subgrafos periféricos nos conjuntos de dados 1, 6, 7, 9, 10, 11 e apresentou um tempo computacional menor em relação ao filtro *LCCFilter*.

A Figura 16 mostra a partição ótima e os tipos de subgrafos periféricos presentes no conjunto de dados 6 (netscience). Os vértices coloridos em verde e azul representam as cliques que a técnica de Arenas et al. [9] consegue filtrar. Em amarelo estão representados cliques maiores que somente o *LCCFilter* e o *BCCFilter* podem filtrar. Finalmente, em roxo estão representados os outros tipos de subgrafos periféricos que somente o *BCCFilter* consegue filtrar. Pode-se observar que os vértices que foram pré-agrupados respeitaram a estrutura de comunidades da partição ótima.

Tabela 2: Tempo computational (segundos), tipo e quantidade de subgrafos periféricos filtrados por *LCCFilter* e *BCCFilter*.

| ID | *LCCFilter* | | *BCCFilter* | |
|---|---|---|---|---|
| | Tempo(s) | Subgrafos | Tempo(s) | Subgrafos |
| 1 | $8,1 \times 10^{-6}$ | $K_1(1)$ | $3,3 \times 10^{-6}$ | $K_1(1)$ $H_5(1)$ |
| 2 | $6,3 \times 10^{-5}$ | $K_1(17)$ $K_2(1)$ | $7,9 \times 10^{-6}$ | $K_1(17)$ $K_2(1)$ |
| 3 | $1,5 \times 10^{-5}$ | $K_1(19)$ | $1,2 \times 10^{-5}$ | $K_1(19)$ |
| 4 | $1,3 \times 10^{-5}$ | $K_1(24)$ $K_2(5)$ $K_3(1)$ | $1,1 \times 10^{-5}$ | $K_1(24)$ $K_2(5)$ $K_3(1)$ |
| 5 | $7,9 \times 10^{-6}$ | $K_1(53)$ | $7,8 \times 10^{-6}$ | $K_1(53)$ |
| 6 | $1,5 \times 10^{-4}$ | $K_1(27)$ $K_2(16)$ $K_3(6)$ $K_4(4)$ | $5,9 \times 10^{-5}$ | $K_1(27)$ $K_2(16)$ $K_3(6)$ $K_4(4)$ $H_3(1)$ $H_4(3)$ $H_5(1)$ $H_6(1)$ $H_8(2)$ $H_9(1)$ $H_{11}(1)$ |
| 7 | $1,3 \times 10^{-4}$ | $K_1(143)$ | $4,5 \times 10^{-5}$ | $K_1(143)$ $H_3(2)$ $H_4(2)$ $H_5(1)$ $H_6(1)$ $H_7(1)$ |
| 8 | $4,6 \times 10^{-5}$ | $K_1(146)$ $K_2(2)$ | $3,1 \times 10^{-5}$ | $K_1(146)$ $K_2(2)$ |
| 9 | $2,4 \times 10^{-4}$ | $K_1(90)$ $K_2(25)$ $K_3(8)$ $K_4(4)$ $K_5(2)$ | $7,3 \times 10^{-5}$ | $K_1(90)$ $K_2(25)$ $K_3(8)$ $K_4(4)$ $K_5(2)$ $H_3(1)$ $H_5(2)$ $H_6(1)$ |
| 10 | $1,3 \times 10^{-4}$ | $K_1(409)$ $K_2(1)$ | $8,2 \times 10^{-5}$ | $K_1(409)$ $K_2(1)$ $H_3(1)$ $H_7(1)$ |
| 11 | $1,2 \times 10^{-4}$ | $K_1(693)$ | $6,6 \times 10^{-5}$ | $K_1(693)$ $H_3(1)$ |

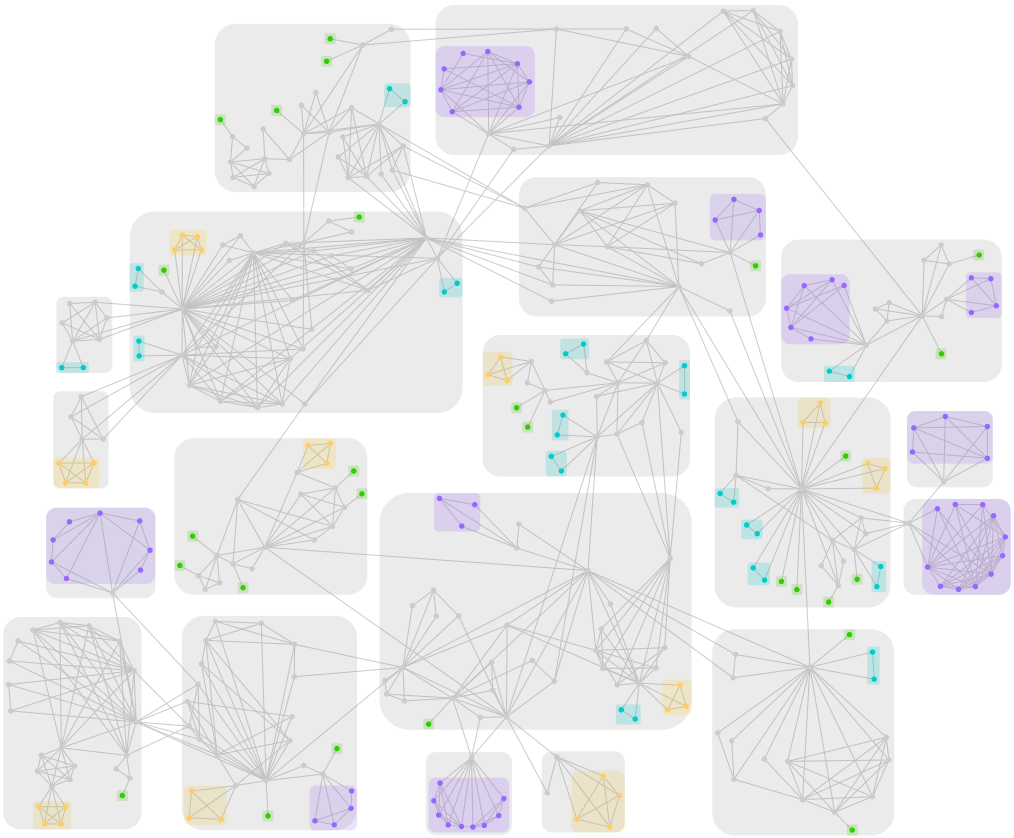


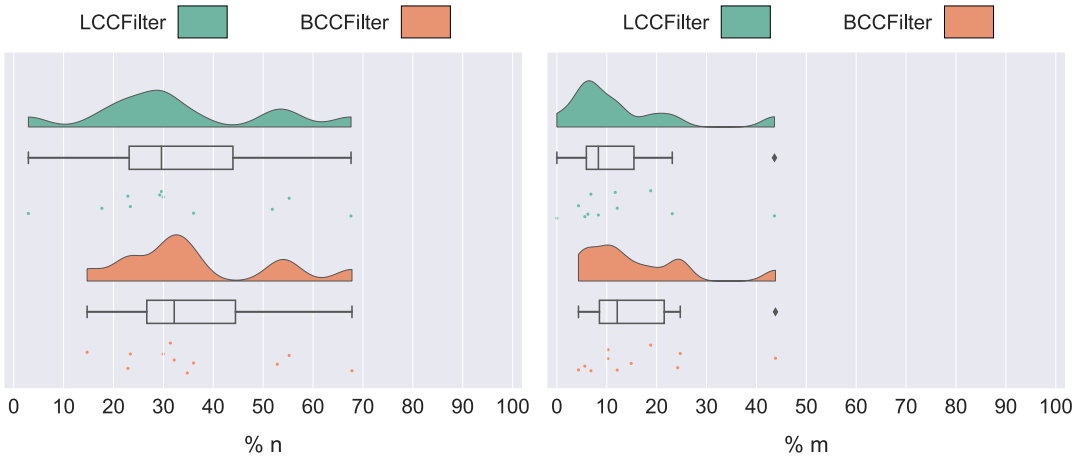
Figura 16: Partição ótima do conjunto de dados 6 (netscience).

A Tabela 3 apresenta o tamanho dos grafos resultantes da aplicação dos filtros propostos neste trabalho. A primeira coluna apresenta o ID de cada conjunto de dados, enquanto que a segunda e terceira colunas representam o tamanho do grafo original do problema. As demais colunas apresentam, para cada filtro, o tamanho do grafo reduzido ($\mathrm{n}'$ e $\mathrm{m}'$). Pode-se observar, que ambos os filtros são capazes de reduzir o tamanho do grafo do problema. Conforme esperado, o filtro *BCCFilter* conseguiu uma maior redução no tamanho dos grafos para os conjuntos de dados 1, 6, 7, 9, 10, 11.

Tabela 3: Tamanho dos grafos após aplicar os filtros *LCCFilter* e *BCCFilter*.

| ID | G | | *LCCFilter* | | *BCCFilter* | |
|---|---|---|---|---|---|---|
| | $\mathrm{n}$ | $\mathrm{m}$ | $\mathrm{n}'$ | $\mathrm{m}'$ | $\mathrm{n}'$ | $\mathrm{m}'$ |
| 1 | 34 | 78 | 33 | 78 | 29 | 70 |
| 2 | 77 | 254 | 59 | 243 | 59 | 243 |
| 3 | 83 | 125 | 64 | 118 | 64 | 118 |
| 4 | 86 | 124 | 55 | 109 | 55 | 109 |
| 5 | 96 | 117 | 43 | 95 | 43 | 95 |
| 6 | 379 | 914 | 312 | 838 | 260 | 688 |
| 7 | 483 | 997 | 340 | 935 | 315 | 895 |
| 8 | 494 | 586 | 346 | 546 | 346 | 546 |
| 9 | 516 | 1 188 | 365 | 1 049 | 350 | 1 011 |
| 10 | 791 | 1 098 | 381 | 844 | 373 | 832 |
| 11 | 1 025 | 1 043 | 332 | 588 | 330 | 586 |

A Figura 17 apresenta a redução do tamanho do grafo do problema em valores percentuais. Os valores percentuais da Figura 17a foram calculados a partir do valores da Tabela 3 através da seguinte fórmula: $100 * (\frac{\mathrm{n}-\mathrm{n}'}{\mathrm{n}})$. Os valores da Figura 17b foram calculados de maneira análoga, alterando-se apenas a fórmula ao contexto de redução de arestas.




(a) Percentual de redução de vértices.

(b) Percentual de redução de arestas.

Figura 17: Percentual de redução do tamanho dos grafos obtidos pelos filtros em relação ao tamanho do grafo original.

Os gráficos da Figura 17 mostram um desempenho superior do filtro *BCCFilter*. A Figura 17a mostra que o *LCCFilter* obteve uma redução do número de vértices entre $2,94\%$ e $67,61\%$, com mediana $29,61\%$. O *BCCFilter* obteve uma redução do número de vértices ligeiramente superior, variando entre $14,71\%$ e $67,8\%$, com mediana $32,17\%$. Na Figura 17b, o *LCCFilter* obteve uma redução do número de arestas entre $0\%$ e $43,62\%$, com mediana $8,32\%$. O *BCCFilter* obteve novamente uma redução superior, variando entre $4,33\%$ e $43,82\%$, com mediana $12,1\%$.

### 4.6.2 *Solução dos modelos matemáticos*

Nesta seção, será feita uma comparação entre os modelos matemáticos criados a partir do grafo original e dos modelos criados a partir dos grafos reduzidos pelos filtros $\mathrm{LCCFilter}$ e $\mathrm{BCCFilter}$. O objetivo é avaliar o tamanho dos modelos matemáticos (variáveis e restrições), o esforço computacional (tempo e memória) necessário para solucioná-los de forma exata, assim como a qualidade da solução obtida.

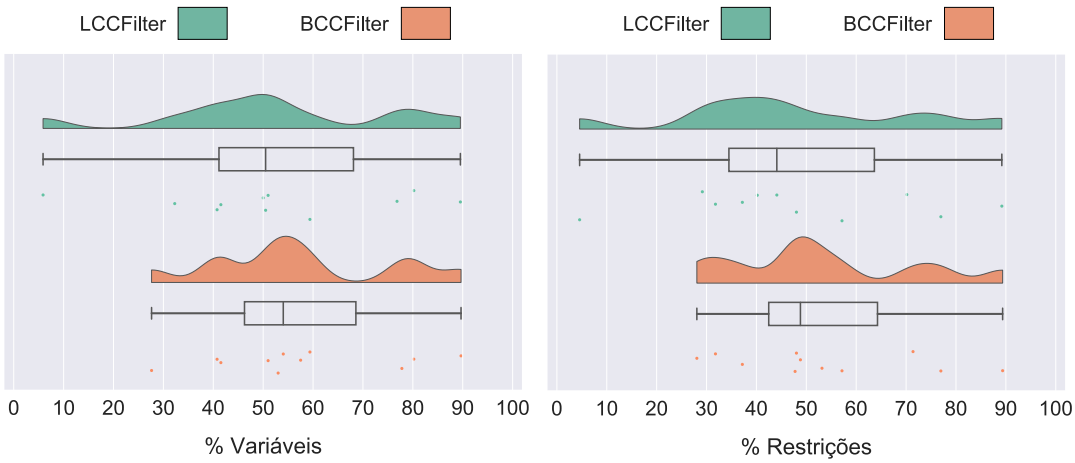
A Tabela 4 apresenta a modularidade e tamanho dos modelos matemáticos (número de variáveis e restrições) criados a partir do grafo original ($\mathrm{G}$) e dos grafos obtidos após a aplicação dos filtros *LCCFilter* e *BCCFilter*. Pode-se observar que os valores de modularidade obtidos pelos modelos que utilizam os grafos reduzidos pelos filtros é igual aos obtidos pelos modelos criados a partir do grafo original. A redução do tamanho do grafo proporcionada pelo filtro *BCCFilter* resultou em modelos matemáticos mais compactos (variáveis e restrições) para os conjuntos de dados 1, 6, 7, 9, 10, 11.

Tabela 4: Modularidade e tamanho dos modelos matemáticos criados a partir do grafo original e dos grafos filtrados por *LCCFilter* e *BCCFilter*.

| ID | Q | #Variáveis | | | #Restrições | | |
|---|---|---|---|---|---|---|---|
| | | G | *LCCFilter* | *BCCFilter* | G | *LCCFilter* | *BCCFilter* |
| 1 | 0,419789 | 561 | 528 | 406 | 4 387 | 4 187 | 3 156 |
| 2 | 0,560008 | 2 926 | 1 711 | 1 711 | 34 685 | 23 657 | 23 657 |
| 3 | 0,530911 | 3 403 | 2 016 | 2 016 | 18 434 | 11 581 | 11 581 |
| 4 | 0,736830 | 3 655 | 1 485 | 1 485 | 20 369 | 8 728 | 8 728 |
| 5 | 0,679669 | 4 560 | 903 | 903 | 21 511 | 4 954 | 4 954 |
| 6 | 0,848580 | 71 631 | 48 516 | 33 670 | 682 739 | 483 639 | 319 951 |
| 7 | 0,778695 | 116 403 | 57 630 | 49 455 | 946606 | 566 353 | 494 600 |
| 8 | 0,859995 | 121 771 | 59 685 | 59 685 | 575 323 | 299 067 | 299 067 |
| 9 | 0,801018 | 132 870 | 66 430 | 61 075 | 1 211 786 | 677 449 | 620 139 |
| 10 | 0,925584 | 312 445 | 72 390 | 69 378 | 1 716 725 | 512 558 | 491 236 |
| 11 | 0,925584 | 524 800 | 54 946 | 54 285 | 2 128 786 | 230 136 | 226 775 |

A Figura 18 apresenta a redução do tamanho do modelo matemático em valores percentuais, calculados a partir do valores da Tabela 4. Os gráficos

da Figura 18 mostram que os grafos filtrados pelo *BCCFilter* dão origem a modelos matemáticos com um menor número de variáveis e restrições que os modelos criados a partir de *LCCFilter*. A Figura 18a mostra que o modelo criado a partir do grafo filtrado pelo *LCCFilter* obteve uma redução do número de variáveis entre $5,88\%$ e $89,53\%$, com mediana $50,49\%$. O *BCCFilter* obteve uma redução superior, variando entre $27,63\%$ e $89,66\%$, com mediana $54,03\%$. Na Figura 18b, o *LCCFilter* obteve uma redução do número de restrições entre $4,56\%$ e $89,19\%$, com mediana $44,09\%$. O *BCCFilter* obteve novamente uma redução superior, variando entre $28,06\%$ e $89,35\%$, com mediana $48,82\%$.




(a) Percentual de redução de variáveis. (b) Percentual de redução de restrições.

Figura 18: Percentual de redução do tamanho dos modelos matemáticos criados a partir dos grafos resultantes da aplicação dos filtros *LCCFilter* e *BCCFilter*.

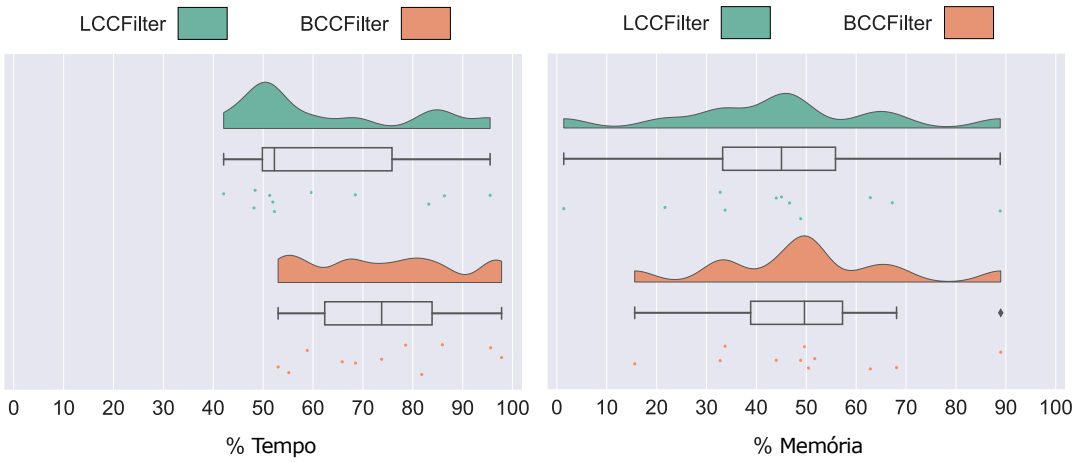
O tempo computacional (segundos) e o consumo de memória (*megabytes*) necessários para solucionar os modelos matemáticos são apresentados na Tabela 5. Pode-se observar os modelos criados a partir dos grafos reduzidos pelos filtros consumiram menos tempo e memória do que os modelos criados a partir do grafo original do problema. Entretanto, os modelos criados a partir dos grafos reduzidos pelo filtro *BCCFilter* obtiveram, como esperado, melhores resultados para os conjuntos de dados 1, 6, 7, 9, 10, 11.

A Figura 19 apresenta a redução do tempo computacional e do consumo de memória obtidos pelos modelos PLI criados a partir dos grafos filtrados por *LCCFilter* e *BCCFilter*. Os valores foram calculados a partir da Tabela 5.

Os gráficos da Figura 19 mostram um desempenho superior dos modelos criado a partir do filtro *BCCFilter*. A Figura 19a mostra que o *LCCFilter* obteve uma redução do tempo computacional entre $42,08\%$ e $95,49\%$, com mediana $52,25\%$. O *BCCFilter* obteve uma redução superior, variando entre $53\%$ e $97,79\%$, com mediana $73,75\%$. Na Figura 19b, o *LCCFilter* obteve uma redução do consumo de memória entre $1,38\%$ e $88,87\%$, com mediana $45,02\%$. O *BCCFilter* obteve novamente uma redução superior, variando entre $15,6\%$ e $88,97\%$, com mediana $49,63\%$.

Tabela 5: Tempo computacional (segundos) e consumo de memória (*megabytes*) dos modelos PLI criados a partir do grafo original e dos grafos filtrados por *LCCFilter* e *BCCFilter*.

| ID | Tempo (s) | | | Memória (MB) | | |
|---|---|---|---|---|---|---|
| | G | *LCCFilter* | *BCCFilter* | G | *LCCFilter* | *BCCFilter* |
| 1 | 0,22 | 0,11 | 0,06 | 1,84 | 1,82 | 1,56 |
| 2 | 0,99 | 0,48 | 0,41 | 7,17 | 4,75 | 4,75 |
| 3 | 2,4 | 1,15 | 1,08 | 4,66 | 3,14 | 3,14 |
| 4 | 0,69 | 0,36 | 0,32 | 4,9 | 2,75 | 2,75 |
| 5 | 0,89 | 0,15 | 0,12 | 5,31 | 1,97 | 1,97 |
| 6 | 44,84 | 25,97 | 15,32 | 118,75 | 93,01 | 57,37 |
| 7 | 11 247,71 | 5 476,34 | 2 411,07 | 187,15 | 102,89 | 94,27 |
| 8 | 91 172,05 | 28 755,83 | 28 717,86 | 119,1 | 60,89 | 60,89 |
| 9 | 23 478,46 | 9 469,66 | 4 274,94 | 219,53 | 117,14 | 108,78 |
| 10 | 28 290,3 | 3 871,47 | 1 246,09 | 313,03 | 102,57 | 99,88 |
| 11 | 2 610,49 | 117,8 | 57,63 | 478,58 | 53,26 | 52,8 |




(a) Percentual de redução de tempo.

(b) Percentual de redução de memória.

Figura 19: Percentual de redução em tempo e memória obtido pelos modelos matemáticos criados a partir dos grafos resultantes da aplicação dos filtros *LCCFilter* e *BCCFilter*.

## 4.7 CONCLUSÕES

Detecção de comunidades é um tópico ativamente estudado e diversas técnicas foram propostas na literatura. A maximização de modularidade via formulação PLI é uma abordagem interessante para se obter a solução exata para esse problema. Essa técnica, entretanto, é fortemente afetada pelo tamanho do grafo do problema, pois os modelos matemáticos criados a partir da formulação PLI tendem a crescer rapidamente à medida que o número de vértices aumenta.

Arenas et al. [9] demostraram que alguns cliques periféricos podem ser filtrados do grafo, criando uma representação menor, que mantém informações relevantes ao contexto de detecção de comunidades. Sua técnica, no entanto, é limitada a cliques compostos por um ou dois vértices.

Este capítulo apresentou dois algoritmos, denotados como $\mathrm{LCCFilter}$ e $\mathrm{BCCFilter}$. Eles estendem os resultados obtidos em Arenas et al. [9]. Os algoritmos propostos foram validados em conjuntos de dados de diferentes domínios de conhecimento. Os resultados experimentais mostram que ambos conseguiram reduzir mais o tamanho do grafo do problema do que a técnica proposta em [9]. Essa diminuição permitiu que fossem criados modelos matemáticos mais compactos (menos variáveis e restrições), e que consomem menos tempo e memória para serem solucionados de forma exata.

Os resultados experimentais mostram que o algoritmo $\mathrm{BCCFilter}$ se sobressaiu sobre o $\mathrm{LCCFilter}$, criando modelos matemáticos que consumiram menos memória e foram solucionados mais rapidamente para alguns conjuntos de dados. Isso foi possível porque esse algoritmo consegue filtrar outros tipos de subgrafos periféricos além de cliques.

A representação reduzida do grafo, obtida pelas técnicas propostas neste capítulo, não alterou a qualidade das soluções obtidas para os conjuntos de dados utilizados nos experimentos. Apesar disso, deve-se reforçar que essas técnicas devem ser utilizadas como heurísticas, pois não foi oferecida uma prova de que são capazes de filtrar o grafo sem alterar a partição ótima do problema. Uma prova formal é prevista como trabalho futuro.

Como trabalho futuro, pretende-se utilizar o algoritmo $\mathrm{BCCFilter}$ em outros contextos, como: agrupamento de dados de *microarray* [57], problema tecnologia de grupos [72], agendamento de aviões [29] e problema de edição de *clusters* [15].

# 5

# PGC E AGRUPAMENTO DE DADOS QUALITATIVOS

Este capítulo aborda uma outra aplicação que a formulação PLI do PGC proposta por Grotschel e Wakabayashi [42] pode ser utilizada: o agrupamento de dados qualitativo. No agrupamento de dados qualitativo o conjunto de dados é composto apenas por atributos categóricos. Esse tipo de conjunto de dados é encontrado em diversas áreas como medicina [23], sociologia [41] e economia [60]. Várias técnicas foram propostas na literatura para solucionar esse problema, a maioria baseada em métodos heurísticos [2].

Pode-se resolver o problema de agrupamentos de dados qualitativos, de forma exata, utilizando-se a formulação PLI proposta por Grotschel e Wakabayashi [42]. Essa formulação, entretanto, cria modelos matemático que demandam um elevado esforço em termos de tempo computacional e consumo de memória para serem solucionados de forma exata.

Trabalhos recentes propuseram formulações que criam modelos matemáticos menos custosos. Miyauchi e Sukegawa [67] fizeram um estudo sobre a formulação original proposta em [42] e identificaram situações em que as restrições dessa formulação podem ser caraterizadas como redundantes. Com base nos resultados desse estudo, os autores propuseram duas novas formulações PLI que evitam que as restrições que eles caracterizaram como redundantes sejam inseridas no modelo matemático. Essas formulações conseguem criar modelos matemáticos mais compactos (menos restrições), que demandam menos esforço computacional (tempo e memória) para serem solucionados de forma exata.

Este capítulo propõe uma nova formulação PLI para o problema de agrupamento de dados qualitativos. Foi realizado, inicialmente, um estudo das duas formulações PLI propostas em [42, 67] para identificar suas principais características e limitações. Durante o estudo foi identificado que um conjunto maior de restrições podem ser caracterizados como redundantes. Os resultados desse estudo foram utilizados para propor uma nova formulação PLI, que evita que essa redundância seja inserida no modelo matemático. Os experimentos computacionais realizados, utilizando conjuntos de dados de diferentes domínios de aplicação, evidenciam que a nova formulações conseguiu criar modelos matemáticos mais compactos (menos restrições) e que demandam menos esforço computacional (tempo e memória) para serem solucionados do que os modelos criados a partir das formulações propostas anteriormente na literatura.

Grotschel e Wakabayashi [42] propuseram uma formulação PLI para esse problema baseada no PGC. A primeira etapa para modelar esse problema como PGC é criar o grafo completo ponderado $K_n = (V(K_n), E(K_n))$, a partir de um conjunto de dados composto por $n$ entidades e $p$ atributos. Neste grafo, cada vértice representará uma entidade e o peso das arestas representará a similaridade entre as entidades. A seguir será apresentado um exemplo que ilustrará como que os autores em [42] propuseram o cálculo da similaridade entre as entidades. A Figura 6 apresenta um conjunto de dados, composto por 6 entidades e 4 atributos (todos categóricos). As explicações que seguem sobre o cálculo da similaridade entre entidades e a criação do grafo completo são feitas tendo como base esse conjunto de dados exemplo.

Tabela 6: Conjunto de dados exemplo.

| Id | Idade | Prescrição | Astigmatismo | Lágrima |
|---|---|---|---|---|
| 1 | jovem | hipermetrope | sim | reduzida |
| 2 | jovem | hipermetrope | sim | normal |
| 3 | pré-presbiopia | hipermetrope | sim | ? |
| 4 | pré-presbiopia | hipermetrope | não | normal |
| 5 | presbiopia | míope | não | reduzida |
| 6 | presbiopia | míope | sim | reduzida |

Os autores em [42] definiram que a similaridade entre duas entidades $i$ e $j$ deve ser calculada a partir da seguinte fórmula:

$$s_{ij} = 2\sum_{k=1}^{p} r_{ij}^{k} - (p - I_{ij}) \tag{14}$$

em que

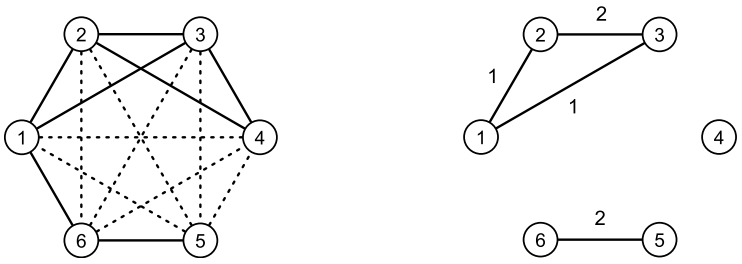
$$r_{ij}^{k} = \begin{cases} 1 & \text{se o atributo k é o mesmo em i e j,} \\ 0 & \text{caso contrário.} \end{cases}$$

$p$ representa o total de atributos do conjunto de dados, e $I_{ij}$ representa o total de atributos incompletos que existirem entre as entidades $i$ e $j$. Esse função define, portanto, valores de similaridade no intervalo $-p \leqslant s_{ij} \leqslant p$.

A matriz de similaridades $S = (s_{ij})_{nxn}$ é obtida aplicando-se a Equação (14) ao conjunto de dados da Tabela 6. A similaridade entre as entidades 1 e 3, por exemplo, é definida como $s_{13} = 2\sum_{k=1}^{4} r_{13}^{k} - (4 - I_{13}) = 2(2) - (4 - 1) = 1$.

$$S = \begin{pmatrix} & 2 & 1 & -2 & -2 & 0 \\ 2 & & 1 & 0 & -4 & -2 \\ 1 & 1 & & 1 & -3 & -1 \\ -2 & 0 & 1 & & -2 & -4 \\ -2 & -4 & -3 & -2 & & 2 \\ 0 & -2 & -1 & -4 & 2 & \end{pmatrix}$$

A Figura 20 ilustra graficamente o grafo completo ponderado do PGC para o exemplo de Tabela 6 . Apesar de não estar representado na Figura 20a, o peso das arestas desse grafo são dados pela matriz de similaridades $S$. A Figura 20b representa uma solução para essa instância no contexto do PGC.




(a) Grafo $K_6$. (b) Solução.

Figura 20: Grafo que representa o exemplo da Tabela 6 e uma possível solução. Arestas com peso negativo foram representadas através de uma linha pontilhada.

Dado que o grafo completo foi construído, pode-se resolver o problema de agrupamento através da seguinte formulação PLI proposta em [42]:

$$(\mathrm{PGC}) \quad \mathit{Maximizar} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} s_{ij} x_{ij}$$

$$\mathit{sujeito\ a}$$

$$x_{ij} + x_{jk} - x_{ik} \leqslant 1 \quad , i < j < k \tag{15}$$

$$x_{ij} - x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k \tag{16}$$

$$-x_{ij} + x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k \tag{17}$$

$$x_{ij} \in \{0, 1\} \quad , i < j$$

As variáveis de decisão $x_{ij}$ assumem o valor 1 se $i$ e $j$ estão no mesmo grupo e 0, caso contrário. O coeficiente $s_{ij}$ representa a similaridade entre as entidades $i$ e $j$. As restrições de transitividade (15-17) garantem que, para todo triângulo do grafo, se duas arestas pertencem à solução então todo o triângulo deve pertencer à solução.

Recentemente, Miyauchi e Sukegawa [67] propuseram duas novas formulações PLI, denominadas neste trabalho como $\mathrm{PGC}_{R1}$ e $\mathrm{PGC}_{R2}$, capazes de

desconsiderar um grande número de restrições redundantes presentes na formulação original $\mathrm{PGC}$. Essas formulações criam modelos matemáticos mais compactos (menos restrições) e demandam menos esforço computacional (tempo e memória) para serem solucionados de forma exata. A formulação $\mathrm{PGC}_{\mathrm{R1}}$ foi introduzida previamente no Capítulo 3, enquanto que a formulação $\mathrm{PGC}_{\mathrm{R2}}$ será descrita mais adiante neste capítulo. Essas formulações, entretanto, sofrem uma forte influência da distribuição do peso das arestas do grafo completo ($\mathrm{K}_n$) que é utilizado como entrada para o PGC. O número de restrições redundantes desconsideradas por essas formulações cai a medida que trabalhamos com grafos que possuam um percentual elevado de arestas positivas.

A limitação das formulações propostas em [67] é explorada na próxima seção, servindo de base para a criação de uma nova formulação, denominada neste trabalho como $\mathrm{PGC}_{\mathrm{R3}}$. Aspectos particulares do problema foram utilizados para amenizar as deficiências das formulações passadas, criando uma formulação que cria modelos matemáticos mais compactos, que demandam menos tempo computacional para serem executados do que os modelos criados a partir das formulações $\mathrm{PGC}_{\mathrm{R1}}$ e $\mathrm{PGC}_{\mathrm{R2}}$.

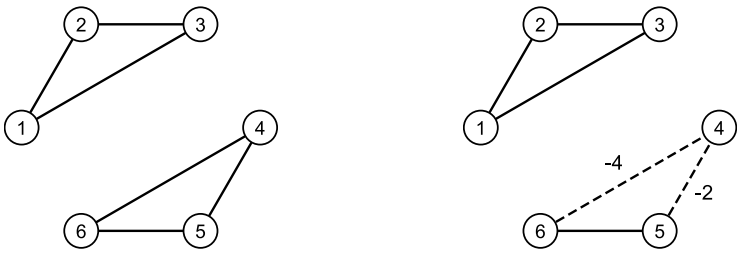
## 5.2 UMA NOVA FORMULAÇÃO PLI PARA O AGRUPAMENTO DE DADOS QUALITATIVOS

Para criarem suas formulações, Miyauchi e Sukegawa [67] analisaram os grupos obtidos na solução ótima e definiram uma condição de suficiência que o peso das arestas dentro desse grupo deve seguir de modo que a estrutura dos grupos seja mantida. A seguinte condição de suficiência foi estabelecida pelos autores para que um grupo da partição ótima seja preservado:

**Condição 1.** Para qualquer subdivisão de um grupo de uma partição ótima em dois subgrupos, deve existir pelo menos uma aresta de peso não negativo no conjunto de arestas entre os subgrupos.

Um exemplo que ilustra a **Condição 1** é apresentado na Figura 21. Suponha a partição viável apresentada na Figura 21a. Se o grupo composto pelos vértices $\{4, 5, 6\}$ for particionado em dois, formando os grupos $A = \{4\}$ e $B = \{5, 6\}$, obteríamos um conjunto $E_{AB}$ composto apenas por arestas de peso negativo (Figura 21b). Haveria uma melhora, portanto, na função objetivo do problema se o vértice $4$ for definido como um novo grupo. O grupo $\{4, 5, 6\}$ da Figura 21a é viável, entretanto, nunca ocorrerá na solução ótima. Os autores provaram em [67] que as restrições de transitividade da formulação $\mathrm{PGC}$ que considerem tal estado podem ser consideradas redundantes e podem ser desconsideradas durante a construção do modelo matemático.

A formulação $\mathrm{PGC}_{\mathrm{R1}}$, proposta em [67], foi criada para impedir a inclusão de restrições de transitividade que não respeitam a **Condição 1**. A formula-




(a) Partição viável. (b) Arestas em $E_{AB}$.

Figura 21: Exemplo em que todas arestas em $E_{AB}$ possuem peso negativo.

ção original PGC é modificada para incluir cláusulas condicionais para cada restrição de transitividade:

$$(PGC_{R1})\ Maximizar \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} s_{ij} x_{ij}$$

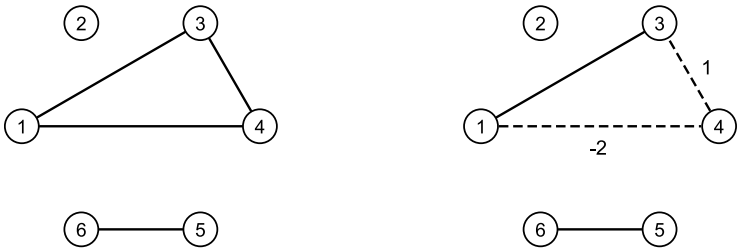
$$\begin{aligned} sujeito\ a \\ x_{ij} + x_{jk} - x_{ik} &\leqslant 1 \quad , i < j < k,\ s_{ij} \geqslant 0 \vee s_{jk} \geqslant 0 \\ x_{ij} - x_{jk} + x_{ik} &\leqslant 1 \quad , i < j < k,\ s_{ij} \geqslant 0 \vee s_{ik} \geqslant 0 \\ -x_{ij} + x_{jk} + x_{ik} &\leqslant 1 \quad , i < j < k,\ s_{jk} \geqslant 0 \vee s_{ik} \geqslant 0 \\ x_{ij} &\in \{0,1\} \qquad , i < j \end{aligned}$$

Em $PGC_{R1}$, cada restrição de transitividade é testada durante a construção do modelo matemático, verificando se há pelo menos uma aresta de peso não negativo no conjunto $E_{AB}$. Apenas as restrições que respeitem essas cláusulas são incluídas no modelo matemático.

Como pode ser observado na seção de resultados experimentais, a formulação $PGC_{R1}$ reduz o número de restrições redundantes e, consequentemente, reduz o tempo computacional necessário para solucionar o modelo matemático. No entanto, essa redução ainda é modesta, justificando-se a busca por formulações alternativas.

Existem condições que $PGC_{R1}$ é incapaz de evitar. Suponha a partição viável apresentada na Figura 22a. Se o grupo composto pelos vértices $\{1,3,4\}$ for particionado em dois, obteremos os grupos $A = \{1,3\}$ e $B = \{4\}$, O conjunto de aresta entre os grupos ($E_{AB}$) respeita a **Condição 1**. No entanto, há uma melhoria na função objetivo do problema se o vértice $4$ for definido como um novo grupo. Isso ocorre porque a soma do peso das arestas no conjunto $E_{AB}$ é negativa. Restrições que consideram tal estado também podem ser consideradas redundantes. Essa condição, entretanto, não foi explorada pelos autores em [67]. Baseado nessa observação, é proposta neste trabalho uma nova condição de suficiência para os grupos de uma partição ótima:

**Condição 2.** Para qualquer subdivisão de um grupo de uma partição ótima em dois subgrupos, a soma do peso das arestas entre os subgrupos deve ser não negativa.




(a) Partição viável. (b) Arestas em $E_{AB}$.

Figura 22: Exemplo em que a soma do peso das arestas em $E_{AB}$ possui valor negativo.

Uma segunda formulação, denominada $PGC_{R2}$ neste trabalho, foi sugerida nas conclusões de [67]. Embora os autores não tenham comentado no texto, ela pode ser vista como uma extensão da formulação $PGC_{R1}$ para evitar que a soma dos peso das aresta em $E_{AB}$ seja negativa, respeitando as **Condições 1 e 2**. Ao apresentarem essa formulação, os autores apenas observaram que essa formulação conseguiu eliminar mais restrições redundantes do que $PGC_{R1}$ e foi capaz de obter a solução ótima para todas as instâncias avaliadas nos experimentos. Os autores deixaram como trabalho futuro a prova de que essa formulação é capaz de manter a solução ótima, assim como fizeram para a formulação $PGC_{R1}$.

Assim como em $PGC_{R1}$, essa nova formulação modifica a formulação original $PGC$ introduzindo cláusulas condicionais para cada restrição de transitividade:

$$
\begin{aligned}
(PGC_{R2})\ Maximizar \quad & \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} s_{ij} x_{ij} \\
sujeito\ a \quad & \\
& x_{ij} + x_{jk} - x_{ik} \leqslant 1 \ , \ i < j < k,\ s_{ij} + s_{jk} \geqslant 0 \\
& x_{ij} - x_{jk} + x_{ik} \leqslant 1 \ , \ i < j < k,\ s_{ij} + s_{ik} \geqslant 0 \\
& -x_{ij} + x_{jk} + x_{ik} \leqslant 1 \ , \ i < j < k,\ s_{jk} + s_{ik} \geqslant 0 \\
& x_{ij} \in \{0, 1\} \qquad , \ i < j
\end{aligned}
$$

A formulação $PGC_{R2}$ testa cada restrição de transitividade, verificando se a soma do peso das arestas em $E_{AB}$ é não-negativa. Restrições que desrespeitem essas cláusulas não são incluídas no modelo matemático. Os modelos matemáticos criados a partir da formulação $PGC_{R2}$ são solucionados mais rapidamente e consomem menos memória que os modelos criados a partir da

formulação $\mathrm{PGC}_{\mathrm{R1}}$. Isso ocorre pois, ao considerar a soma do peso das arestas em $\mathrm{E}_{AB}$, a formulação $\mathrm{PGC}_{\mathrm{R2}}$ restringe a intensidade do peso das arestas negativas que podem pertencer ao conjunto $\mathrm{E}_{AB}$.

Embora as formulações $\mathrm{PGC}_{\mathrm{R1}}$ e $\mathrm{PGC}_{\mathrm{R2}}$ crie modelos matemáticos mais compactos que a formulação original $\mathrm{PGC}$, ainda há espaço para melhorias em termos de tempo computacional e consumo de memória. Essas formulações se concentraram apenas no papel das arestas do conjunto $\mathrm{E}_{AB}$. Pode-se expandir, portanto, a análise de quais restrições de transitividade devem ser incluídas no modelo matemático, para preservar os grupos na partição ótima, ao se estudar a distribuição do peso das arestas dos triângulos considerados pelas restrições de transitividade do modelo $\mathrm{PGC}$.

A Figura 23 ilustra os tipos de triângulos que podem ser encontrados no grafo completo que serve de entrada para o PGC. Pode-se observar, na Figura 23, que todos os triângulos do tipo T4 (todas arestas com peso estritamente negativo) e os triângulos T1 que possuem todas arestas com peso estritamente positivo são casos extremos em que não há dúvida sobre a decisão de manter os vértices juntos ou separados. Neste trabalho, foi conjecturado que as restrições que envolvem esses casos são redundantes. Não será necessário, portanto, criar restrições de transitividade que verifiquem triângulos T1 que possuam todas as arestas com peso estritamente positivo, pois existe um consenso de que os vértices devem estar juntos. Não é necessário garantir a transitividade nesse caso, pois a otimização da função objetivo do problema definirá o valor das variáveis correspondentes a essas arestas como 1 durante a otimização do modelo matemático. Da mesma forma, restrições que considerem triângulos do tipo T4 podem ser desconsideradas, uma vez que existe um consenso de que os vértices devem ser separados. Consequentemente, as restrições de transitividade que devem ser incluídas serão as referentes aos triângulos T1 (pesos não estritamente positivos), T2 e T3, onde o consenso não está presente.

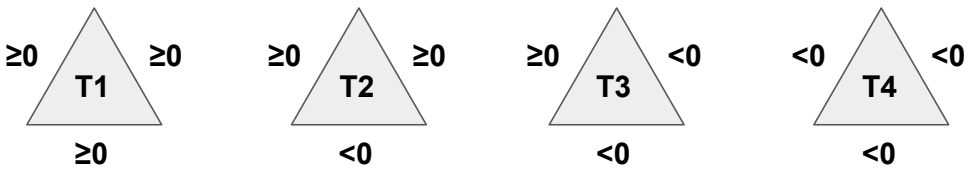



Figura 23: Tipos de triângulos que podem ser encontrados em $\mathrm{K}_{n}$.

A formulação $\mathrm{PGC}_{\mathrm{R3}}$, proposta neste trabalho, pode eliminar as restrições de transitividade correspondentes aos valores extremos dos triângulos T1 e T4 e controlar a distribuição do peso das arestas negativas nos triângulos T2 e T3. A formulação $\mathrm{PGC}$ é modificada para incluir cláusulas condicionais para cada restrição de transitividade:

$$(\mathrm{PGC}_{R3})\ \text{Maximizar} \quad \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} s_{ij}x_{ij}$$

$$\text{sujeito a}$$

$$x_{ij} + x_{jk} - x_{ik} \leqslant 1\ ,\ i < j < k,\ (s_{ij} + s_{jk} \geqslant 0) \wedge (s_{ij} \geqslant 0) \wedge (s_{ik} \leqslant 0)$$
$$x_{ij} - x_{jk} + x_{ik} \leqslant 1\ ,\ i < j < k,\ (s_{ij} + s_{ik} \geqslant 0) \wedge (s_{ij} \geqslant 0) \wedge (s_{jk} \leqslant 0)$$
$$-x_{ij} + x_{jk} + x_{ik} \leqslant 1\ ,\ i < j < k,\ (s_{ik} + s_{jk} \geqslant 0) \wedge (s_{ik} \geqslant 0) \wedge (s_{ij} \leqslant 0)$$
$$x_{ij} \in \{0, 1\},\ i < j$$

As cláusulas condicionais de $\mathrm{PGC}_{R3}$ respeitam as **Condições 1 e 2** apresentadas anteriormente. Quando aplicada à restrição (15) da formulação PGC, a cláusula condicional $(s_{ij} + s_{jk} \geqslant 0) \wedge (s_{ij} \geqslant 0) \wedge (s_{ik} \leqslant 0)$ assegura que pelo menos uma das arestas em $E_{AB}$ seja não-negativa, fixando $(s_{ij} \geqslant 0)$, e que a soma dos pesos das arestas em $E_{AB}$ seja não-negativa, utilizando $(s_{ij} + s_{jk} \geqslant 0)$.

No contexto dos triângulos da Figura 23, a primeira cláusula condicional da formulação $\mathrm{PGC}_{R3}$ evita a criação de restrições de transitividade correspondente aos triângulos T4, ao utilizar $(s_{ij} \geqslant 0)$, e aos triângulos T1 estritamente positivos, ao utilizar $(s_{ik} \leqslant 0)$. O termo $(s_{ij} + s_{jk} \geqslant 0)$ controla os pesos das arestas negativas no contexto dos triângulos T2 e T3.

A próxima seção apresenta o conjunto de experimentos que foram realizados para comparar os modelos matemáticos criados a partir formulações apresentadas nesta seção. Serão comparados o número de restrições redundantes que foram eliminadas, assim como o desempenho da técnica que resolve esses modelos de forma exata em termos de tempo computacional e consumo de memória.

## 5.3 RESULTADOS EXPERIMENTAIS

Nos experimentos, os algoritmos utilizados foram codificados em C++ e executados em um computador com a seguinte configuração: Intel Core i7-6770HQ (3,5GHz) com 32 GB de RAM executando o Windows 10 de 64 bits. O *solver* comercial IBM ILOG CPLEX [48] 12.7.1 foi utilizado para resolver os modelos matemáticos, e a linguagem R [83] foi empregada na análise estatística. O código fonte está disponível em `https://github.com/LuizHNLorena/QualitativeClustering/`.

A figura 24 apresenta a metodologia utilizada para conduzir os experimentos computacionais. Primeiro, nas etapas 1 e 2, os conjuntos de dados foram padronizados da seguinte forma: o símbolo "?" foi utilizado para representar valores de atributos ausentes, enquanto que o atributo de classe foi removido (quando fornecido). Na etapa 3, quatro diferentes modelos matemáticos foram criados para cada conjunto de dados: $\mathrm{PGC}$ é o modelo composto por todas as restrições; $\mathrm{PGC}_{R1}$ e $\mathrm{PGC}_{R2}$ são os modelos criados a partir das formulações

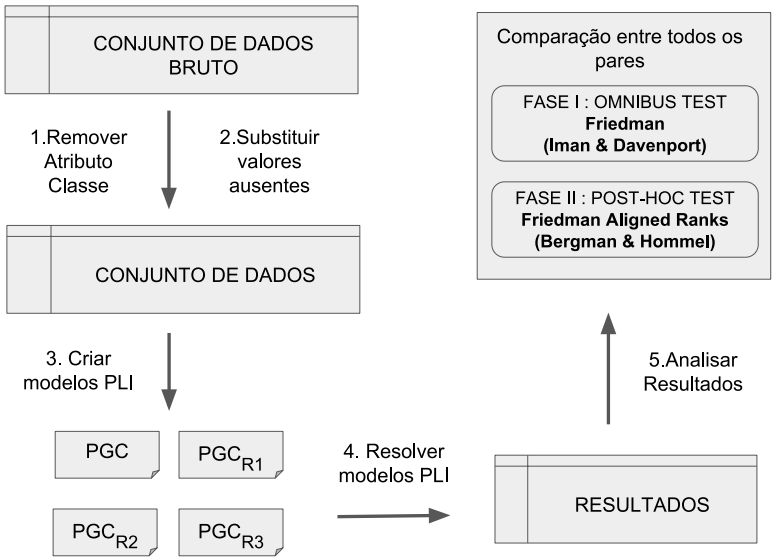



Figura 24: Metodologia utilizada nos experimentos computacionais.

propostas por Miyauchi e Sukegawa [67]; e $\mathrm{PGC}_{\mathrm{R3}}$ é o modelo criado pela formulação proposta neste trabalho. Na etapa 4, os modelos matemáticos foram resolvidos pelo *solver* da Programação Linear Inteira. Por fim, os resultados obtidos foram analisados no passo 5, utilizando testes estatísticos, que verificaram se existem diferenças significativas entre o desempenho obtido pelo *solver* na resolução dos diferentes modelos matemáticos.

A Tabela 7 apresenta os conjuntos de dados utilizados para avaliar os modelos PLI. Cada conjunto de dados recebeu um ID único (primeira coluna) e um nome (segunda coluna). As colunas restantes possuem os seguintes significados: $\mathrm{n}$ representa o número de entidades; $\mathrm{p}$ o número de atributos; $\mathrm{Incompleto}$ informa se o conjunto de dados tem entidades com atributos ausentes; e, finalmente, $\%\mathrm{Positivos}$ revela o percentual de arestas com peso positivo presentes no grafo $\mathrm{K_n}$.

Os conjuntos de dados cujo ID pertence ao conjunto {2-7,9,13,14,17,18} são exemplos clássicos do PGC propostos na literatura. Essas instâncias podem ser encontradas no Apêndice de [42]. Conjuntos de dados cujo ID pertence ao conjunto {1,3,8,10-12,15,16,19,20} representam instâncias de problemas de diferentes domínios de conhecimento selecionados do repositório de conjuntos de dados para experimentos de aprendizado de máquina da *UCI* [61]. Todos os conjuntos de dados são compostos apenas por atributos categóricos.

As subseções que seguem detalham os resultados obtidos após a aplicação das etapas 1-5 da metodologia apresentada na Figura 24. Os modelos são comparados nos seguintes aspectos: eliminação de restrições, tempo computacional necessário para sua solução, eficiência de espaço e robustez.

Tabela 7: Conjuntos de dados utilizados nos experimentos.

| ID | Nome | n | p | Incompleto | %Positivos |
|---|---|---|---|---|---|
| 1 | Lenses | 24 | 4 | Não | 60,87 |
| 2 | Wildcats | 30 | 14 | Não | 62,76 |
| 3 | Lung Cancer | 32 | 56 | Sim | 69,15 |
| 4 | Cars | 33 | 13 | Não | 72,73 |
| 5 | Workers | 34 | 13 | Não | 58,82 |
| 6 | Cetacea | 36 | 16 | Sim | 27,94 |
| 7 | Micro | 40 | 14 | Não | 28,59 |
| 8 | Soybean (Small) | 47 | 35 | Não | 99,17 |
| 9 | UNO | 54 | 3 | Não | 40,81 |
| 10 | Sponge | 76 | 45 | Sim | 65,33 |
| 11 | Zoo | 101 | 16 | Não | 71,23 |
| 12 | Bridges | 108 | 13 | Sim | 47,87 |
| 13 | UNO1b | 139 | 3 | Não | 47,72 |
| 14 | UNO2b | 145 | 15 | Não | 75,64 |
| 15 | Lymphography | 148 | 18 | Não | 67,46 |
| 16 | Teaching Evaluation | 151 | 5 | Não | 8,12 |
| 17 | UNO1a | 158 | 3 | Não | 39,73 |
| 18 | UNO2a | 158 | 15 | Não | 64,45 |
| 19 | Hayes-Roth | 160 | 3 | Não | 24,01 |
| 20 | Primary Tumor | 339 | 17 | Sim | 93,87 |

### 5.3.1 *Eliminação de restrições e tempo computacional*

A Tabela 8 fornece resultados detalhados sobre o número de restrições e o tempo computacional necessário para solucionar cada modelo matemático. A primeira coluna faz referência ao ID do conjunto de dados da Tabela 7, o valor da função objetivo é apresentado na coluna $\mathrm{Obj}$, enquanto que as colunas $\#\mathrm{Restrições}$ e $\mathrm{Tempo}$ representam o total de restrições dos modelos e o tempo computacional necessário para resolvê-los, respectivamente. Os melhores resultados foram destacados em negrito.

Pode-se observar, a partir da Tabela 8, que todos os modelos atingem o valor ótimo para a função objetivo. No entanto, a formulação $\mathrm{PGC}_{\mathrm{R3}}$ criou modelos matemáticos com um menor número de restrições, e que demandaram menos tempo computacional para serem solucionados pelo *solver* do que os modelos criados a partir das demais formulações.

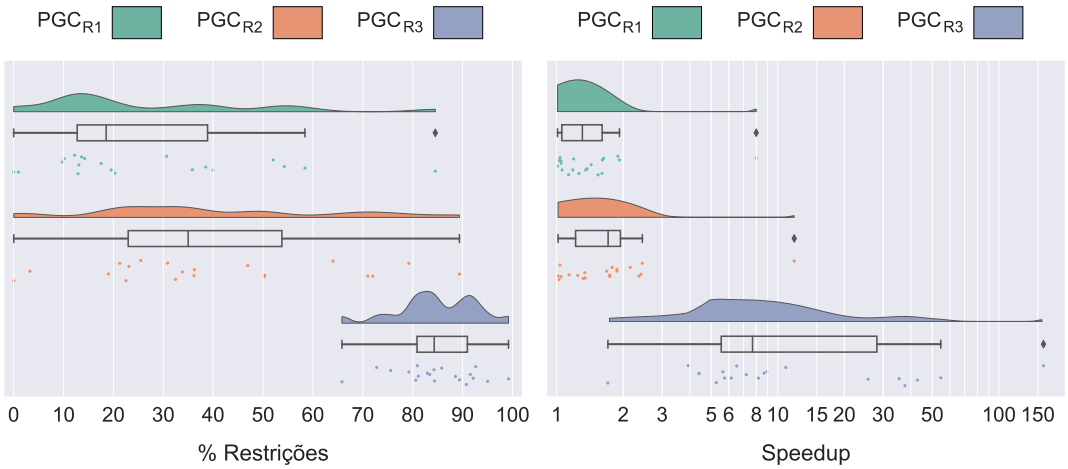
O gráfico da Figura 25 complementa os resultados apresentados na Tabela 8. A Figura 25a mostra a redução do número de restrições, em termos percentuais, que as formulações $\mathrm{PGC}_{\mathrm{R1}}$, $\mathrm{PGC}_{\mathrm{R2}}$ e $\mathrm{PGC}_{\mathrm{R3}}$ obtiveram em relação à formulação original $\mathrm{PGC}$. Os modelos criados a partir da formulação $\mathrm{PGC}_{\mathrm{R2}}$ foram mais compactos (menos restrições) que os modelos criados a partir da formulação $\mathrm{PGC}_{\mathrm{R1}}$. A formulação $\mathrm{PGC}_{\mathrm{R3}}$, entretanto, criou modelos ainda

Tabela 8: Valor da função objetivo, número de restrições dos modelos matemáticos e tempo computacional (segundos) necessário para solucioná-los.

| ID | Obj | #Restrições | | | | Tempo (s) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | PGC | $PGC_{R1}$ | $PGC_{R2}$ | $PGC_{R3}$ | PGC | $PGC_{R1}$ | $PGC_{R2}$ | $PGC_{R3}$ |
| 1 | 72 | 6 072 | 5 208 | 3 024 | **2 076** | 0,38 | 0,37 | 0,28 | **0,22** |
| 2 | 1 304 | 12 180 | 10 043 | 8 060 | **1 905** | 0,16 | 0,14 | 0,13 | **0,02** |
| 3 | 3 472 | 14 880 | 12 959 | 11 089 | **2 809** | 0,31 | 0,30 | 0,27 | **0,02** |
| 4 | 1 501 | 16 368 | 14 708 | 13 257 | **2 324** | 0,42 | 0,41 | 0,40 | **0,05** |
| 5 | 964 | 17 952 | 14 444 | 12 413 | **3 422** | 0,43 | 0,36 | 0,32 | **0,05** |
| 6 | 967 | 21 420 | 9 798 | 6 214 | **1 980** | 0,138 | 0,10 | 0,08 | **0,02** |
| 7 | 406 | 29 640 | 14 217 | 6 161 | **3 147** | 0,20 | 0,12 | 0,08 | **0,04** |
| 8 | 14 613 | 48 645 | 48 638 | 48 630 | **395** | 2,86 | 2,70 | 2,79 | **0,01** |
| 9 | 798 | 74 412 | 45 756 | 36 945 | **12 703** | 1,93 | 1,20 | 1,11 | **0,21** |
| 10 | 25 677 | 210 900 | 182 196 | 162 169 | **43 798** | 2,95 | 2,34 | 2,31 | **0,59** |
| 11 | 16 948 | 499 950 | 451 130 | 387 355 | **136 133** | 106,89 | 56,50 | 45,33 | **20,30** |
| 12 | 3 867 | 612 468 | 393 126 | 220 328 | **96 487** | 305,37 | 294,93 | 125,12 | **54,12** |
| 13 | 11 775 | 1 313 967 | 910 908 | 838 054 | **254 913** | 57,56 | 35,73 | 33,19 | **2,24** |
| 14 | 71 818 | 1 492 920 | 1 310 497 | 1 175 303 | **117 649** | 34,11 | 28,59 | 25,36 | **0,79** |
| 15 | 19 174 | 1 588 188 | 1 380 477 | 1 072 877 | **387 728** | 4359,76 | 3043,93 | 2327,78 | **989,90** |
| 16 | 1 108 | 1 687 425 | 261 201 | 179 647 | **84 337** | 23,07 | 2,88 | 1,94 | **0,61** |
| 17 | 12 197 | 1 934 868 | 1 161 623 | 1 026 713 | **321 261** | 99,06 | 51,52 | 46,04 | **2,79** |
| 18 | 72 820 | 1 934 868 | 1 542 583 | 1 235 377 | **142 490** | 53,03 | 38,62 | 29,77 | **0,97** |
| 19 | 2 800 | 2 009 760 | 836 313 | 563 121 | **229 952** | 553,72 | 359,68 | 295,67 | **89,83** |
| 20 | 323 614 | 19 307 067 | 19 121 209 | 18 680 264 | **1 634 216** | 612,82 | 607,38 | 592,64 | **70,63** |

mais compactos que as demais, apresentando $100\%$ de sua distribuição acima das medianas de $\mathrm{PGC_{R1}}$ e $\mathrm{PGC_{R2}}$. De acordo com quartil inferior do *boxplot* de $\mathrm{PGC_{R3}}$, $75\%$ das instâncias obtiveram uma porcentagem de eliminação de restrição acima de $80\%$.

A Figura 25b apresenta o ganho em termos de tempo computacional (*Speedup*) obtido pelo *solver* na solução dos modelos mais compactos frente ao modelo original criado a partir da formulação $\mathrm{PGC}$. O ganho é calculado através da razão entre o tempo computacional obtido pelo modelo $\mathrm{PGC}$ e o tempo computacional do modelo considerado. O eixo horizontal do gráfico está em escala logarítmica. Pode-se concluir que o *solver* teve um desempenho um pouco melhor nos modelos criados a partir da formulação $\mathrm{PGC_{R2}}$ do que nos modelos criados a partir da formulação $\mathrm{PGC_{R1}}$. O desempenho do *solver* foi melhor em todas os modelos criados a partir da formulação $\mathrm{PGC_{R3}}$. O quartil inferior de $\mathrm{PGC_{R3}}$ mostra que $75\%$ das instâncias obtiveram um ganho de tempo computacional acima de $5$.




(a) Percentual de restrições eliminadas. (b) *Speedup*.

Figura 25: Percentual de restrições eliminadas e *speedup* obtido pelos modelos matemáticos criados a partir das formulações $\mathrm{PGC_{R1}}$, $\mathrm{PGC_{R2}}$, $\mathrm{PGC_{R3}}$ frente ao modelo criado a partir da formulação $\mathrm{PGC}$.

### 5.3.2 *Eficiência de espaço*

A Tabela 9 mostra a quantidade de memória em *megabytes* necessária para armazenar cada modelo matemático. Pode-se observar que as formulações $\mathrm{PGC_{R1}}$ e $\mathrm{PGC_{R2}}$ criam modelos que consomem menos memória que os modelos criados a partir da formulação $\mathrm{PGC}$. A formulação $\mathrm{PGC_{R3}}$, entretanto, cria modelos que possuem o menor consumo de memória do que as demais. Para o conjunto de dados 20, por exemplo, o modelo criado a partir da formulação $\mathrm{PGC_{R3}}$ requer 960 MB de memória, enquanto os modelos restantes exigem um pouco mais do que 10 GB.

Tabela 9: Consumo de memória, em *megabytes*, dos modelos matemáticos.

| ID | PGC | $\text{PGC}_{\text{R1}}$ | $\text{PGC}_{\text{R2}}$ | $\text{PGC}_{\text{R3}}$ |
|---|---|---|---|---|
| 1 | 4,8 | 4,2 | 3,0 | **2,5** |
| 2 | 8,5 | 7,1 | 6,2 | **2,5** |
| 3 | 9,8 | 8,8 | 8,0 | **3,0** |
| 4 | 11,2 | 9,7 | 9,1 | **2,7** |
| 5 | 12,2 | 9,7 | 8,6 | **3,4** |
| 6 | 13,7 | 7,1 | 5,0 | **2,6** |
| 7 | 18,6 | 9,7 | 5,1 | **3,3** |
| 8 | 30,4 | 30,4 | 30,4 | **1,9** |
| 9 | 46,1 | 29,3 | 23,7 | **9,0** |
| 10 | 123,1 | 111,5 | 95,3 | **29,2** |
| 11 | 282,2 | 262,5 | 229,6 | **85,7** |
| 12 | 357,0 | 232,2 | 131,7 | **61,0** |
| 13 | 746,8 | 525,9 | 482,3 | **159,7** |
| 14 | 884,1 | 745,8 | 690,9 | **73,4** |
| 15 | 923,0 | 838,7 | 649,6 | **232,3** |
| 16 | 963,4 | 163,0 | 114,1 | **54,5** |
| 17 | 1 092,0 | 686,2 | 573,9 | **189,8** |
| 18 | 1 092,1 | 905,0 | 716,1 | **91,4** |
| 19 | 1 122,5 | 712,9 | 277,3 | **192,9** |
| 20 | 11 174,4 | 11 098,7 | 10 920,3 | **961,9** |

A Figura 26 resume os resultados apresentados na Tabela 9. A quantidade de redução de consumo de memória é calculada para cada modelo utilizando o modelo PGC como base.

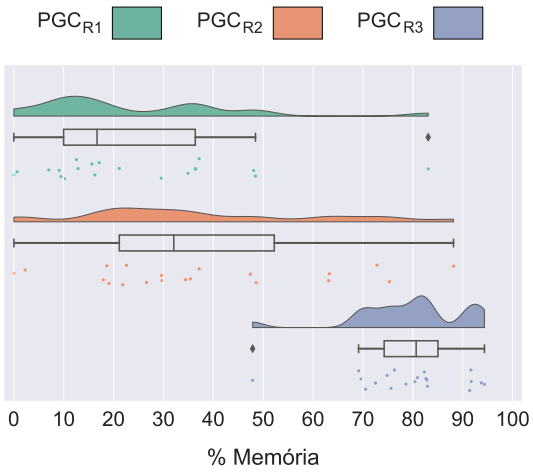



Figura 26: Percentual de redução do consumo de memória dos modelos matemáticos criados a partir das formulações $\text{PGC}_{\text{R1}}$, $\text{PGC}_{\text{R2}}$, $\text{PGC}_{\text{R3}}$ frente ao modelo criado a partir da formulação PGC.

O gráfico da Figura 26 mostra que os modelos criados a partir das formulações $\mathrm{PGC}_{\mathrm{R1}}$ e $\mathrm{PGC}_{\mathrm{R2}}$ são capazes de reduzir o consumo de memória, mas os modelos criados a partir da formulação $\mathrm{PGC}_{\mathrm{R3}}$ consomem menos memória do que os demais. O quartil inferior de $\mathrm{PGC}_{\mathrm{R3}}$ mostra uma redução de consumo memória acima de $75\%$ para $75\%$ das instâncias.

### 5.3.3 *Robustez*

Em [67], os autores mencionaram que a formulação $\mathrm{PGC}_{\mathrm{R1}}$ criaria modelos matemáticos mais compactos se o grafo $K_n$ tivesse uma pequena proporção de arestas de peso positivo. Esse comportamento pode ser observado no gráfico da Figura 27, em que eixo horizontal inferior lista os conjuntos de dados listados em ordem crescente em relação à proporção de arestas com peso positivo ($\%\mathrm{Positivos}$ da Tabela 7), enquanto que o eixo horizontal superior lista o ID dos conjuntos de dados (ID da Tabela 7). O eixo vertical representa o percentual de restrições da formulação $\mathrm{PGC}$ que foram eliminadas pelas formulações $\mathrm{PGC}_{\mathrm{R1}}$, $\mathrm{PGC}_{\mathrm{R2}}$ e $\mathrm{PGC}_{\mathrm{R3}}$.

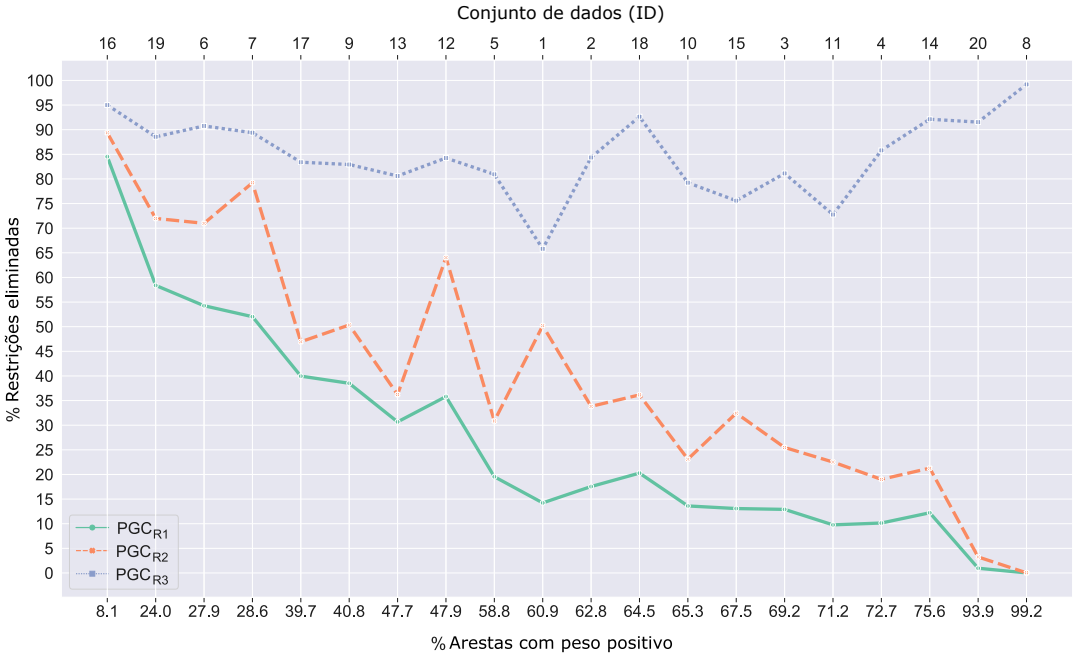



Figura 27: Percentual de restrições de transitividades eliminadas do modelo matemático a medida que se aumenta o número de arestas com peso positivo no grafo completo do problema.

A Figura 27 mostra que, a medida que a proporção de arestas com peso positivo aumenta no grafo de entrada do problema, menos restrições redundantes são encontradas pelas condicionais inseridas na formulação $\mathrm{PGC}_{\mathrm{R1}}$. O mesmo comportamento pode ser observado para a formulação $\mathrm{PGC}_{\mathrm{R2}}$. No caso da formulação $\mathrm{PGC}_{\mathrm{R3}}$ não se observa esse comportamento. Portanto, o *coeficiente de correlação de Spearman* [79] ($\rho$) foi utilizado para quantificar

essa influência. O valor do coeficiente obtido para $\mathrm{PGC}_{\mathrm{R1}}$ foi de $\rho = -0,97$, demonstrando que existe uma forte correlação negativa entre o aumento do percentual de arestas de peso positivo no grafo e a capacidade da formulação de identificar restrições redundantes. A formulação $\mathrm{PGC}_{\mathrm{R2}}$ também é afetada pelo aumento da proporção de arestas com peso positivo. Seu coeficiente de correlação é $\rho = -0,95$, o que representa uma forte correlação negativa.

A formulação proposta nesta tese ($\mathrm{PGC}_{\mathrm{R3}}$) apresentou algumas flutuações, mas não foi diretamente afetada pela proporção de arestas de peso positivo (Figura 27). Obteve-se um coeficiente de correlação igual a $\rho = -0,058$, o que representa uma fraca correlação negativa entre as variáveis analisadas. Essa formulação criou modelos matemáticos mais compactos para todas as instâncias analisadas, obtendo o pior desempenho no conjunto de dados com ID 1 ($65,81\%$ de restrições eliminadas). Esse resultado, entretanto, é superior ao obtido por $\mathrm{PGC}_{\mathrm{R1}}$ e $\mathrm{PGC}_{\mathrm{R2}}$, que eliminaram, respectivamente, $14,23\%$ e $50,19\%$ das restrições.

### 5.3.4 *Significância estatística*

Finalmente, seguindo o passo 5 da metodologia proposta na Figura 24, um teste pareado e não paramétrico foi utilizado para assegurar que existe um diferença significativa entre o percentual de restrições eliminadas por cada formulação. Este trabalho seguiu as recomendações de Demsar [25] e Garcia et al. [37, 36] para realizar essas comparações.

O teste de Friedman [35] com correção de Iman e Davenport [49] foi aplicado como teste *omnibus* para detectar se pelo menos uma das formulações tem um desempenho diferente das demais. O Teste de Rank Alinhado de Friedman [45] com a correção de Bergman e Hommel [14] foi empregado como teste *post-hoc* para detectar qual delas teve melhor desempenho.

Este trabalho utilizou o pacote em linguagem R fornecido por Calvo e Guzman [20], que contém os testes *omnibus* e *post-hoc*, necessários para realização dos testes de hipótese. O nível de significância de $\alpha = 0,05$ foi considerado.

O teste *omnibus* obteve um *p-valor* igual a $2,2 \times 10^{-16}$, mostrando que pelo menos um das formulações teve um desempenho diferente. O teste *post-hoc* mostrou que todas as formulações tiveram um desempenho diferente, e que a formulação $\mathrm{PGC}_{\mathrm{R3}}$ superou todos as demais em relação a capacidade de remover restrições.

## 5.4 CONCLUSÕES

O agrupamento de dados categóricos pode ser solucionado por meio da formulação de Programação Linear Inteira do PGC proposta por Grotschel e Wakabayashi [42]. Essa formulação, entretanto, produz modelos matemáticos

compostos por um grande número de variáveis e restrições. Recentemente, formulações que dão origem a modelos mais compactos foram criados por Miyauchi e Sukegawa [67]. Embora essas formulações consigam evitar que um grande número de restrições redundantes sejam introduzidas no modelo matemático, elas possuem limitações.

Neste capítulo, um nova formulação, denominada $\mathrm{PGC}_{\mathrm{R3}}$, foi proposta. Resultados experimentais mostraram que os modelos matemáticos criados a partir dela foram mais compactos (menor número de restrições), e demandaram menos esforço computacional (tempo e memória) para serem solucionados de forma exata. A nova formulação se mostrou robusta que as formulações previamente propostas na literatura, conseguindo criar modelos mais compactos, independente da configuração do peso das arestas do grafo utilizado como entrada para o problema.

Como trabalho futuro, pretende-se estudar o impacto que a seleção de atributos pode ter sobre os modelos criados a partir da formulação $\mathrm{PGC}_{\mathrm{R3}}$. Como a matriz de similaridade é criada a partir da similaridade dos atributos do conjunto de dados, a seleção de um subconjunto de atributos menor irá mudar seus valores. Essa nova configuração da matriz pode ser benéfica para resolver problemas que se mostraram computacionalmente custosos, como o conjunto de dados 15 utilizado nos experimentos.

Espera-se que os resultados obtidos no contexto do método exato possam ser utilizados para orientar a construção de melhores heurísticas para este problema. Além disso, pode-se tentar utilizar a formulação $\mathrm{PGC}_{\mathrm{R3}}$ em outros contextos, como agrupamento de dados de *microarray* [57], detecção de comunidades [28, 66], agrupamento de máquinas e peças [72] e agendamento de aviões [29].

# 6

# PGC E PROBLEMA DE EDIÇÃO DE CLUSTERS

Neste capítulo, estuda-se o Problema de Edição de *Clusters* (PEC) [17]. Esse problema foi previamente investigado em contextos diversos, como bioinformática [13, 15], agrupamento de documentos [10], segmentação de imagens [71], consenso entre agrupamentos [3, 39], e agrupamento de dados compostos apenas por atributos qualitativos [39, 42].

O Problema de Edição de *Clusters* (PEC) [17] pode ser resolvido de forma ótima por meio da formulação PLI proposta por Grotschel e Wakabayashi [42] apresentada no Capítulo 3. Essa abordagem, entretanto, cria modelos matemáticos com um grande número de restrições.

Este capítulo apresenta um estudo dos triângulos correspondentes às restrições de transitividade do modelo PGC aplicado ao contexto do PEC. Esse estudo identificou quais restrições de transitividade do PGC podem ser consideradas redundantes no contexto do PEC. Os resultados foram utilizados para propor uma nova formulação que evita essa redundância, criando modelos matemáticos mais compactos. Os experimentos computacionais realizados mostram que os modelos criados a partir da nova formulação resolvem problemas com até $2000$ vértices.

## 6.1 O PROBLEMA DE EDIÇÃO DE CLUSTERS

O Problema de Edição de *Clusters* (PEC) tem como objetivo efetuar o menor número possível de edições (inserção ou remoção) de arestas para particionar um grafo não-direcionado em uma união de cliques disjuntas. A Figura 28 mostra uma representação gráfica deste problema em que são necessárias três edições para que o grafo seja particionado em cliques disjuntas, representadas pelos vértices $\{1,2,3,4\}$ e $\{5,6,7\}$.

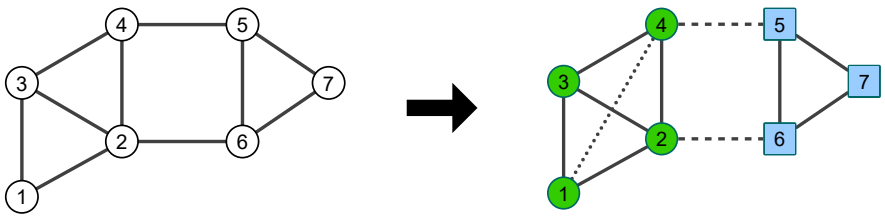



Figura 28: Exemplo de Edição de *Clusters*. Linhas sólidas, tracejadas e pontilhadas representam, respectivamente, arestas que foram mantidas, removidas e inseridas.

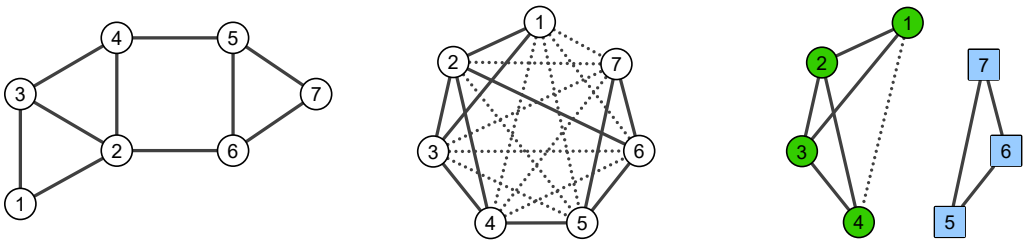
Como é um problema NP-difícil [10, 78], foram propostas para sua solução heurísticas, algoritmos de aproximação e métodos capazes de reduzir o tamanho do problema [12, 32, 43, 74].

A resolução deste problema de forma exata foi apresentada no trabalho de Bocker et al. [17], quando os autores utilizaram a formulação PLI proposta por Grotschel e Wakabayashi [42]. Os autores propuseram técnicas para reduzir o tamanho do problema por meio de um pré-processamento que identifica padrões que possam ser removidos sem afetar os grupos da solução ótima. O problema reduzido é posteriormente resolvido de forma exata utilizando a técnica de *Planos de Corte* [87] proposta em [42].

## 6.2 EDIÇÃO DE CLUSTERS VIA PLI

O PEC pode ser formulado como um problema de maximização ou minimização [21]. Este trabalho considera a versão em que o objetivo é minimizar o número de edições de arestas para particionar um grafo não direcionado e não ponderado $G = (V, E)$, composto por $n = |V|$ vértices e $m = |E|$ arestas.

A formulação PLI para o PEC é uma variante da formulação PGC apresentada no Capítulo 3. O PEC é adaptado ao contexto do PGC da seguinte forma (Figura 29): cria-se um grafo completo ponderado $K_n = (V(K_n), E(K_n))$ utilizando o grafo $G$ como base. Os pesos das arestas ($w_{ij}$) são definidos como 1 se existe uma aresta entre os vértices $i$ e $j$, e $-1$ caso contrário.




(a) Grafo não direcionado e não ponderado $G$.

(b) Grafo ponderado $K_n$. Linhas sólidas representam $w_{ij} = 1$, e linhas pontilhadas $w_{ij} = -1$.

(c) Solução ótima.

Figura 29: Adaptando o PEC ao contexto do PGC.

Definidos o grafo $K_n$ e o peso associado a cada uma de suas arestas, a seguinte formulação PLI pode ser considerada para o PEC:

$$
\begin{aligned}
&(\mathrm{PEC}_{\mathrm{ILP}}) \\
&\text{Minimizar} \quad |E(G)| - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_{ij} \\
&\text{sujeito a} \\
&\qquad x_{ij} + x_{jk} - x_{ik} \leqslant 1 \qquad , i < j < k \qquad (18) \\
&\qquad x_{ij} - x_{jk} + x_{ik} \leqslant 1 \qquad , i < j < k \qquad (19) \\
&\qquad -x_{ij} + x_{jk} + x_{ik} \leqslant 1 \qquad , i < j < k \qquad (20) \\
&\qquad x_{ij} \in \{0, 1\} \qquad , i, j \in [1..n]
\end{aligned}
$$

em que $x_{ij} = 1$ se $i$ e $j$ fazem parte da solução e $0$, caso contrário. Na função objetivo, $m$ representa o número de arestas do grafo $G$, enquanto que a segunda parcela da equação considera a otimização das edições de arestas. A diferença entre esses termos resultará no número mínimo de edições necessárias para particionar o grafo $G$.

A segunda parcela da função objetivo da formulação $\mathrm{PEC}_{\mathrm{ILP}}$ admite as seguintes possibilidades:

1. $w_{ij} = 1$ e $x_{ij} = 1$ (nenhuma edição)
2. $w_{ij} = 1$ e $x_{ij} = 0$ (remove aresta)
3. $w_{ij} = -1$ e $x_{ij} = 1$ (insere aresta)
4. $w_{ij} = -1$ e $x_{ij} = 0$ (nenhuma edição)

Portanto, as edições de arestas são consideradas na solução da formulação $\mathrm{PEC}_{\mathrm{ILP}}$ apenas nos casos 2 e 3.

A formulação $\mathrm{PEC}_{\mathrm{ILP}}$ utiliza as mesmas restrições de transitividade do modelo PGC. As restrições (18-20) garantem que, para todo triângulo do grafo $K_n$, se duas arestas pertencem ao mesmo clique então todo o triângulo pertence ao clique. Por se basear na formulação PGC, a formulação $\mathrm{PEC}_{\mathrm{ILP}}$ apresenta a mesma limitação: uma grande quantidade de restrições de transitividade ($O(n^3)$).

## 6.3 UMA NOVA FORMULAÇÃO PARA O PEC

Uma nova formulação para o PEC é proposta neste trabalho, tendo $\mathrm{PEC}_{\mathrm{ILP}}$ como base. Inicialmente, identifica-se a redundância das restrições de transitividade do modelo $\mathrm{PEC}_{\mathrm{ILP}}$ dentro do contexto do PEC. Essa informação é utilizada posteriormente para criar uma formulação que evite que tais redundâncias sejam introduzidas no modelo matemático.

A redundância é identificada por meio do comportamento de $\mathrm{PEC}_{\mathrm{ILP}}$ frente aos possíveis triângulos presentes no grafo $K_n$ (Figura 30). A análise desses

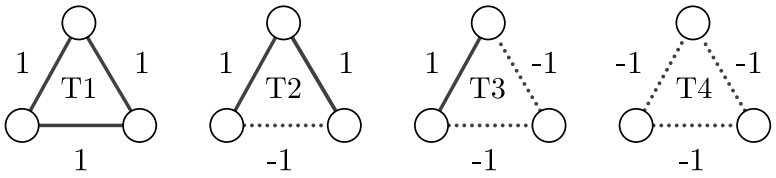



Figura 30: Tipos de triângulos que podem existir em $K_n$.

triângulos ajudará a identificar quais restrições não resultarão em edições e que poderão, consequentemente, ser desconsideradas pela nova formulação.

As restrições de transitividade relacionadas a triângulos T1, T3 e T4 podem ser desconsideradas devido as seguintes razões:

- T1: durante a solução de $PEC_{ILP}$ serão atribuídos os valores $x_{ij} = x_{jk} = x_{ik} = 1$ para $w_{ij} = w_{jk} = w_{ik} = 1$, pois todas as restrições de transitividade serão satisfeitas.
- T3: durante a solução de $PEC_{ILP}$ será atribuído o valor 1 para as varáveis relativas as arestas de peso positivo e 0 para as variáveis restantes, pois isso irá satisfazer todas as restrições de transitividade.
- T4: durante a solução de $PEC_{ILP}$ serão atribuídos os valores $x_{ij} = x_{jk} = x_{ik} = 0$ para $w_{ij} = w_{jk} = w_{ik} = -1$ pois todas as restrições de transitividade serão satisfeitas.

Triângulos to tipo T2 devem ser considerados, pois as restrições de transitividade não são satisfeitas quando variáveis correspondente a arestas de peso positivo possuem o valor 1. Para satisfazer essas restrições, deve-se atribuir 1 para a variável correspondente a aresta de peso negativo. Entretanto, isso resultaria em a uma edição de aresta. Existe outra possibilidade: a remoção de arestas do grafo.

Existem três variantes do triângulo T2, dependendo da ordem considerada para seus vértices (Figura 31). Considerando a otimização da função objetivo de $CE_{ILP}$, as melhores possibilidades de edição (1 edição), que irão satisfazer todas as restrições de transitividade, são as seguintes:

- T2A:

  $x_{ij} = x_{jk} = x_{ik} = 1$ (1 aresta inserida)

  $x_{ij} = 1, x_{jk} = x_{ik} = 0$ (1 aresta removida)

  $x_{jk} = 1, x_{ij} = x_{ik} = 0$ (1 aresta removida)

- T2B:

  $x_{ij} = x_{jk} = x_{ik} = 1$ (1 aresta inserida)

  $x_{ij} = 1, x_{jk} = x_{ik} = 0$ (1 aresta removida)

  $x_{ik} = 1, x_{ij} = x_{jk} = 0$ (1 aresta removida)

- T2C:

  $x_{ij} = x_{jk} = x_{ik} = 1$ (1 aresta inserida)

  $x_{ik} = 1, x_{ij} = x_{jk} = 0$ (1 aresta removida)

  $x_{jk} = 1, x_{ij} = x_{ik} = 0$ (1 aresta removida)

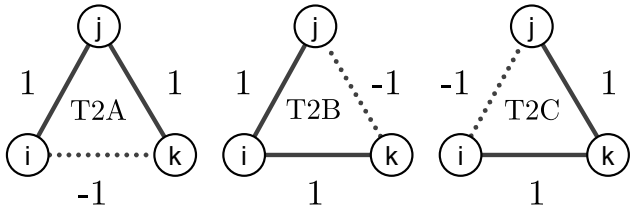



Figura 31: Possíveis configurações para o peso das arestas de triângulos do tipo T2.

Triângulos T2 também são conhecidos como "triplas conflitantes" (*conflict triples*) na literatura e são a raiz de diversas técnicas de redução de dados [17, 15].

Neste trabalho, a informação do peso das arestas dos triângulos é utilizada diretamente para identificar quais restrições de transitividade devem ser consideradas na formulação $\mathrm{PEC}_{\mathrm{ILP}}$ durante a construção do modelo matemático. Apenas as restrições relacionadas com triângulos do tipo T2 são consideradas. Desta maneira, uma nova formulação, denominada $\mathrm{PECR}_{\mathrm{ILP}}$, é proposta. Ela modificada a formulação $\mathrm{PEC}_{\mathrm{ILP}}$, incluindo cláusulas condicionais para as restrições de transitividade:

$(\mathrm{PECR}_{\mathrm{ILP}})$

$$\text{Minimizar} \quad |E(G)| - \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_{ij}$$

sujeito a

$$\begin{aligned}
x_{ij} + x_{jk} - x_{ik} &\leqslant 1 \quad , \; i < j < k \; , \; w_{ij} = 1 \wedge w_{jk} = 1 \wedge w_{ik} = -1 \\
x_{ij} - x_{jk} + x_{ik} &\leqslant 1 \quad , \; i < j < k \; , \; w_{ij} = 1 \wedge w_{jk} = -1 \wedge w_{ik} = 1 \\
-x_{ij} + x_{jk} + x_{ik} &\leqslant 1 \quad , \; i < j < k \; , \; w_{ij} = -1 \wedge w_{jk} = 1 \wedge w_{ik} = 1 \\
x_{ij} &\in \{0, 1\} \quad , \; i, j \in [1..n]
\end{aligned}$$

O testes condicionais permitirão que apenas as restrições de transitividade correspondentes aos triângulos T2A, T2B e T2C sejam inseridas no modelo.

## 6.4 RESULTADOS EXPERIMENTAIS

Os experimentos e algoritmos foram codificado na linguagem C++ e executados em um computador com a seguinte configuração: processador Intel Core i7-6770HQ (3,5 GHz) e 32 GB de memória RAM. O *solver* comercial IBM ILOG

CPLEX 12.7.1 [48] foi utilizado para resolver os modelos matemáticos criados a partir das formulações $\mathrm{PEC}_{\mathrm{ILP}}$ e $\mathrm{PECR}_{\mathrm{ILP}}$.

Os conjuntos de dados que seguem foram utilizados:

- Grafos LFR. Grafos criados com a ferramenta desenvolvida por Lancichinetti et al. [59]. Os seguintes parâmetros foram utilizados: número de vértices $n \in \{50, 100, 200\}$, grau médio igual a 5 e grau máximo igual a 10. Valores crescente do parâmetro de mistura ($\mu$) foram utilizados. Os valores padrão foram utilizados para os parâmetros restantes da ferramenta.

- Grafos aleatórios não-ponderados. Foram criados por Bocker et al. [17] da seguinte maneira: dado um número de vértices $n$ e um parâmetro $k$, seleciona-se uniformemente um inteiro $i \in \{1, 2, ..., n\}$, que definirá um grupo composto por $i$ vértices. O processo continua com os $n = n - i$ vértices restantes até que $n \leqslant 5$. Os vértices que ainda não tiverem grupo serão atribuídos a um novo grupo. Finalmente, um valor estimado $k' \leqslant k$ é utilizado para realizar edições (adição/remoção de arestas) no grafo obtido. Os conjuntos de dados de tamanho $n \in \{100, 200, 300, 1000, 1500, 2000\}$ foram selecionados para os experimentos.

### 6.4.1 *Experimentos com grafos LFR*

A Tabela 10 apresenta o desempenho obtido pelo *solver* na resolução dos modelos matemáticos criados a partir das formulações $\mathrm{PEC}_{\mathrm{ILP}}$ e $\mathrm{PECR}_{\mathrm{ILP}}$, no contexto dos grafos LFR. A coluna $n$ representa o número de vértices do grafo e $\mu$ é o parâmetro de mistura. As colunas $\mathrm{Obj}$, $\#\mathrm{R}$ e $\mathrm{Tempo}$ são definidas para ambas as formulações e representam, respectivamente, o valor da função objetivo, o número de restrições do modelo matemático, e o tempo computacional em segundos necessário para solucioná-los. Finalmente, as duas últimas colunas comparam o desempenho de $\mathrm{PECR}_{\mathrm{ILP}}$ e de $\mathrm{PEC}_{\mathrm{ILP}}$. A coluna $\%\mathrm{R}$ representa o percentual de restrições que $\mathrm{PECR}_{\mathrm{ILP}}$ conseguiu remover da formulação original, enquanto que $S$ representa o ganho em tempo computacional (*speedup*) de $\mathrm{PECR}_{\mathrm{ILP}}$ frente a $\mathrm{PEC}_{\mathrm{ILP}}$. Esse valor é calculado através da razão entre o tempo obtido por $\mathrm{PEC}_{\mathrm{ILP}}$ e o obtido por $\mathrm{PECR}_{\mathrm{ILP}}$.

É possível observar, a partir da Tabela 10, que os modelos criados a partir de $\mathrm{PECR}_{\mathrm{ILP}}$ obtêm a solução ótima para todos os conjunto de dados. Além disso, o modelo matemático é mais compacto e demanda menos esforço computacional para ser solucionado. A coluna $\%\mathrm{R}$ demonstra que o $\mathrm{PECR}_{\mathrm{ILP}}$ é capaz de reduzir o tamanho do modelo em mais de $99\%$ para todos os conjunto de dados. Essa redução de tamanho garante um ganho em tempo computacional que varia de $15$ a $13755$ vezes o tempo para executar o modelo da formulação original. Todas as instâncias foram resolvidas pelo $\mathrm{PECR}_{\mathrm{ILP}}$ em menos de $2$ segundos.

Tabela 10: Desempenho obtido pelo *solver* na resolução dos modelos criados a partir das formulações $\mathrm{PEC}_{\mathrm{ILP}}$ e $\mathrm{PECR}_{\mathrm{ILP}}$ nos grafos LFR.

| | | $\mathrm{PEC}_{\mathrm{ILP}}$ | | | $\mathrm{PECR}_{\mathrm{ILP}}$ | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | $\mu$ | Obj | #R | Tempo (s) | Obj | #R | Tempo (s) | %R | S |
| 50 | 0,1 | 50 | 58 800 | 0,63 | 50 | 250 | 0,04 | 99,57 | 15,18 |
| | 0,2 | 65 | 58 800 | 1,01 | 65 | 375 | 0,07 | 99,36 | 15,27 |
| | 0,3 | 71 | 58 800 | 1,21 | 71 | 411 | 0,07 | 99,30 | 17,28 |
| | 0,4 | 80 | 58 800 | 1,79 | 80 | 486 | 0,10 | 99,17 | 18,77 |
| | 0,5 | 78 | 58 800 | 2,70 | 78 | 423 | 0,11 | 99,28 | 24,81 |
| | 0,6 | 74 | 58 800 | 1,68 | 74 | 381 | 0,07 | 99,35 | 24,96 |
| | 0,7 | 88 | 58 800 | 2,73 | 88 | 533 | 0,14 | 99,09 | 19,90 |
| | 0,8 | 87 | 58 800 | 5,37 | 87 | 522 | 0,21 | 99,11 | 25,02 |
| | 0,9 | 89 | 58 800 | 5,29 | 89 | 528 | 0,27 | 99,10 | 19,74 |
| 100 | 0,1 | 86 | 485 100 | 6,23 | 86 | 428 | 0,05 | 99,91 | 118,44 |
| | 0,2 | 108 | 485 100 | 6,52 | 108 | 627 | 0,05 | 99,87 | 126,95 |
| | 0,3 | 147 | 485 100 | 11,73 | 147 | 893 | 0,15 | 99,82 | 75,86 |
| | 0,4 | 149 | 485 100 | 16,01 | 149 | 890 | 0,11 | 99,82 | 143,08 |
| | 0,5 | 173 | 485 100 | 46,76 | 173 | 1078 | 0,23 | 99,78 | 201,69 |
| | 0,6 | 183 | 485 100 | 82,62 | 183 | 1153 | 0,63 | 99,76 | 131,68 |
| | 0,7 | 182 | 485 100 | 74,43 | 182 | 1127 | 0,36 | 99,77 | 208,65 |
| | 0,8 | 193 | 485 100 | 72,91 | 193 | 1260 | 0,27 | 99,74 | 267,89 |
| | 0,9 | 172 | 485 100 | 49,43 | 172 | 1004 | 0,19 | 99,79 | 257,55 |
| 200 | 0,1 | 223 | 3 940 200 | 157,55 | 223 | 1118 | 0,18 | 99,97 | 865,92 |
| | 0,2 | 223 | 3 940 200 | 74,58 | 223 | 1 270 | 0,11 | 99,97 | 651,88 |
| | 0,3 | 257 | 3 940 200 | 96,49 | 257 | 1 563 | 0,11 | 99,96 | 911,65 |
| | 0,4 | 296 | 3 940 200 | 221,62 | 296 | 1 688 | 0,17 | 99,96 | 1 310,29 |
| | 0,5 | 331 | 3 940 200 | 558,39 | 331 | 2 004 | 0,17 | 99,95 | 3 460,32 |
| | 0,6 | 333 | 3 940 200 | 956,43 | 333 | 1 925 | 0,20 | 99,95 | 4 697,06 |
| | 0,7 | 342 | 3 940 200 | 5 074,88 | 342 | 1 908 | 0,37 | 99,95 | 13 755,17 |
| | 0,8 | 376 | 3 940 200 | 10 547,42 | 376 | 2 333 | 1,73 | 99,94 | 6 084,59 |
| | 0,9 | 481 | 3 940 200 | 5 855,49 | 481 | 3 388 | 2,03 | 99,91 | 2 877,47 |

Instâncias com tamanho $n \geqslant 300$ não foram testadas, pois os modelos criados com a formulação $\mathrm{PEC}_{\mathrm{ILP}}$ não conseguem ser solucionados devido a falta de memória.

### 6.4.2 *Experimentos com grafos aleatórios não-ponderados*

Os experimentos foram realizados nos conjuntos de dados propostos em [17]. Esse conjunto de dados é composto por grafos criados com um limite superior do número de edições $k = c * n$, em que $c \in \{0,25\ ,\ 0,5\ ,\ 1,25\ ,\ 1,5\ ,\ 1,75\ ,\ 2\}$. Foram criados 10 grafos para cada par $(n, k)$, sendo $n \in \{100, 200, 300, 1000, 1500, 2000\}$. Os experimentos foram realizados em duas etapas. Os conjuntos de dados de tamanho $n \in \{100, 200, 300\}$ foram considerados inicialmente, pois os modelos criados a partir da formulação $\mathrm{PEC}_{\mathrm{ILP}}$ não são capazes de resolver instâncias com $n > 300$, devido a falta de memória. Neste primeiro experimento, comparou-se o desempenho do *solver* para resolver os modelos criados a partir das formulações $\mathrm{PEC}_{\mathrm{ILP}}$ e $\mathrm{PECR}_{\mathrm{ILP}}$, em termos de consumo de tempo e qualidade da solução. Posteriormente, foram realizados experimentos com grafos de tamanho $n \in \{1000, 1500, 2000\}$ para testar a escalabilidade dos modelos criado a partir da formulação $\mathrm{PECR}_{\mathrm{ILP}}$.

A Tabela 11 apresenta os resultados obtidos para o $\mathrm{PEC}_{\mathrm{ILP}}$ e $\mathrm{PECR}_{\mathrm{ILP}}$ considerando os grafos com $n \leqslant 300$. Cada linha representa os valores médios para os 10 grafos gerados a partir dos pares $(n, k)$. A coluna $n$ representa o número de vértices do grafo, e $k$ representa o limite superior para o número de edições. As colunas $\mathrm{Obj}$, #R e $\mathrm{Tempo}$ são definidas para ambas as formulações e representam, respectivamente, o valor da função objetivo, o número de restrições do modelo matemático, e o tempo computacional necessário para solucioná-los (segundos).

Finalmente, a coluna %R representa o percentual de restrições que a formulação $\mathrm{PECR}_{\mathrm{ILP}}$ conseguiu remover em relação ao número de restrições da formulação original, enquanto que $S$ representa o ganho em tempo computacional (*speedup*) obtido pelo *solver*, calculado através da razão que relaciona o tempo necessário para resolver o modelo criado a partir da formulação $\mathrm{PEC}_{\mathrm{ILP}}$ e o tempo obtido pelo modelo criado a partir da formulação $\mathrm{PECR}_{\mathrm{ILP}}$.

É possível verificar, a partir da Tabela 11, que os modelos criados a partir da formulação $\mathrm{PECR}_{\mathrm{ILP}}$ conseguem resultados melhores que os criados com a formulação $\mathrm{PEC}_{\mathrm{ILP}}$. Os modelos obtidos possuem um menor número de restrições, e são solucionados em menor tempo. Todas as instâncias foram resolvidas em menos de 2 segundos, obtendo-se um percentual de redução de restrições (%R) frente ao modelo original acima de 97%. Os modelos criados com a formulação $\mathrm{PECR}_{\mathrm{ILP}}$ foram resolvidos mais rapidamente, obtendo ganhos que variaram entre $43,14$ e $11386,81$ frente ao tempo de execução dos modelos obtidos por $\mathrm{PEC}_{\mathrm{ILP}}$.

A Tabela 12 mostra o resultado dos experimentos realizados com os grafos com $n > 300$. O objetivo foi testar a escalabilidade dos modelos matemáti-

Tabela 11: Desempenho obtido pelo *solver* na resolução dos modelos criados a partir das formulações $\mathrm{PEC}_{\mathrm{ILP}}$ e $\mathrm{PECR}_{\mathrm{ILP}}$ nos grafos aleatórios com até 300 vértices.

| | | $\mathrm{PEC}_{\mathrm{ILP}}$ | | | $\mathrm{PECR}_{\mathrm{ILP}}$ | | | | |
|---|---|---|---|---|---|---|---|---|---|
| n | k | Obj | #R | Tempo (s) | Obj | #R | Tempo (s) | %R | S |
| 100 | 25 | 25,0 | 485 100 | 10,85 | 25,0 | 1 745,8 | 0,03 | 99,64 | 346,66 |
| | 50 | 49,0 | 485 100 | 7,03 | 49,0 | 3 001,4 | 0,04 | 99,38 | 164,36 |
| | 75 | 74,2 | 485 100 | 5,35 | 74,2 | 4 409,1 | 0,07 | 99,09 | 84,59 |
| | 100 | 97,8 | 485 100 | 5,14 | 97,8 | 6 547,8 | 0,11 | 98,65 | 49,88 |
| | 125 | 121,4 | 485 100 | 4,79 | 121,4 | 7 726,2 | 0,09 | 98,41 | 52,59 |
| | 150 | 145,6 | 485 100 | 4,92 | 145,6 | 8 369,7 | 0,10 | 98,27 | 52,64 |
| | 175 | 168,2 | 485 100 | 4,61 | 168,2 | 10 108,1 | 0,10 | 97,92 | 47,19 |
| | 200 | 192,4 | 485 100 | 4,62 | 192,4 | 11 542,6 | 0,11 | 97,62 | 43,14 |
| 200 | 50 | 50,0 | 3 940 200 | 242,98 | 50,0 | 5 889,6 | 0,08 | 99,85 | 3266,11 |
| | 100 | 99,6 | 3 940 200 | 141,69 | 99,6 | 13 819,6 | 0,22 | 99,65 | 728,56 |
| | 150 | 149,6 | 3 940 200 | 64,90 | 149,6 | 17 697,5 | 0,23 | 99,55 | 299,32 |
| | 200 | 198,0 | 3 940 200 | 51,93 | 198,0 | 24 434,4 | 0,30 | 99,38 | 188,05 |
| | 250 | 248,0 | 3 940 200 | 41,75 | 248,0 | 32 242,4 | 0,36 | 99,18 | 121,56 |
| | 300 | 295,2 | 3 940 200 | 41,19 | 295,2 | 36 497,3 | 0,40 | 99,07 | 105,71 |
| | 350 | 343,0 | 3 940 200 | 41,09 | 343,0 | 37 213,1 | 0,41 | 99,06 | 103,98 |
| | 400 | 391,4 | 3 940 200 | 41,41 | 391,4 | 46 879,4 | 0,50 | 98,81 | 85,44 |
| 300 | 75 | 74,8 | 13 365 300 | 1 621,70 | 74,8 | 12 447,7 | 0,15 | 99,91 | 11 386,81 |
| | 150 | 149,4 | 13 365 300 | 875,95 | 149,4 | 29 056,6 | 0,42 | 99,78 | 2 265,54 |
| | 225 | 224,0 | 13 365 300 | 430,22 | 224,0 | 45 402,3 | 0,70 | 99,66 | 683,91 |
| | 300 | 298,2 | 13 365 300 | 262,33 | 298,2 | 57 886,7 | 0,79 | 99,57 | 340,42 |
| | 375 | 371,5 | 13 365 300 | 203,77 | 371,5 | 78 641,0 | 0,98 | 99,41 | 215,10 |
| | 450 | 445,2 | 13 365 300 | 188,92 | 445,2 | 80 117,3 | 0,98 | 99,40 | 206,10 |
| | 525 | 518,8 | 13 365 300 | 186,77 | 518,8 | 103 056,6 | 1,19 | 99,23 | 165,03 |
| | 600 | 593,2 | 13 365 300 | 195,56 | 593,2 | 126 468,7 | 1,46 | 99,05 | 140,61 |

cos criados a partir da formulação $\mathrm{PECR_{ILP}}$. Não foi possível criar modelos a partir da formulação $\mathrm{PEC_{ILP}}$, pois eles excedem a capacidade de memória da máquina. Portanto, apenas o número de restrições é reportado. Cada linha representa a média dos resultados obtidos para os 10 grafos criados a partir do para $(n, k)$. A coluna $n$ representa o número de vértices do grafo; $k$ representa o limite superior para o número de edições; apenas o número médio de restrições é apresenta para a formulação $\mathrm{PEC_{ILP}}$. As colunas $\mathrm{Obj}$, #R e $\mathrm{Tempo}$ representam os valores médios da função objetivo, número de restrições do modelo matemático e o tempo computacional necessário para solucioná-lo (segundos), respectivamente. Finalmente, a coluna $\%\mathrm{R}$ destaca o percentual médio de remoção de restrições obtido por $\mathrm{PECR_{ILP}}$ frente a formulação original.

Tabela 12: Desempenho obtido pelo *solver* na resolução dos modelos criados a partir das formulações $\mathrm{PEC_{ILP}}$ e $\mathrm{PECR_{ILP}}$ nos grafos aleatórios com até 2000 vértices.

| | | $\mathrm{PEC_{ILP}}$ | $\mathrm{PECR_{ILP}}$ | | | |
|---|---|---|---|---|---|---|
| $n$ | $k$ | #R | Obj | #R | Tempo (s) | %R |
| 1 000 | 250 | 498 501 000 | 249,8 | 162941,7 | 9,29 | 99,97 |
| | 500 | 498 501 000 | 499,8 | 338 462,5 | 14,56 | 99,93 |
| | 750 | 498 501 000 | 749,0 | 475 923,0 | 17,91 | 99,90 |
| | 1 000 | 498 501 000 | 998,0 | 652 948,9 | 21,47 | 99,87 |
| | 1 250 | 498 501 000 | 1 247,5 | 900 571,3 | 25,61 | 99,82 |
| | 1 500 | 498 501 000 | 1 494,6 | 1 040 558,8 | 24,14 | 99,79 |
| | 1 750 | 498 501 000 | 1 743,2 | 1 235 538,8 | 32,12 | 99,75 |
| | 2 000 | 498 501 000 | 1 990,0 | 1 183 563,7 | 41,57 | 99,76 |
| 1 500 | 375 | 1 684 126 500 | 374,8 | 363 423,8 | 15,95 | 99,98 |
| | 750 | 1 684 126 500 | 749,6 | 717 436,6 | 39,40 | 99,96 |
| | 1 125 | 1 684 126 500 | 1 123,6 | 1 167 055,5 | 73,22 | 99,93 |
| | 1 500 | 1 684 126 500 | 1 494,6 | 1 040 558,4 | 24,14 | 99,79 |
| | 1 875 | 1 684 126 500 | 1 872,8 | 1 817 375,0 | 89,43 | 99,89 |
| | 2250 | 1 684 126 500 | 2 245,2 | 2 125 008,0 | 150,52 | 99,87 |
| | 2625 | 1 684 126 500 | 2 618,6 | 2 276 352,3 | 227,35 | 99,86 |
| | 3 000 | 1 684 126 500 | 2 991,2 | 3 048 967,0 | 386,80 | 99,82 |
| 2 000 | 500 | 3 994 002 000 | 500,0 | 739 933,5 | 76,19 | 99,98 |
| | 1 000 | 3 994 002 000 | 998,8 | 1 149 414,0 | 77,29 | 99,97 |
| | 1 500 | 3 994 002 000 | 1 498,4 | 2 138 914,5 | 220,12 | 99,95 |
| | 2 000 | 3 994 002 000 | 1 998,0 | 2 708 588,2 | 340,46 | 99,93 |
| | 2 500 | 3 994 002 000 | 2 497,2 | 3 210 726,6 | 403,91 | 99,92 |
| | 3 000 | 3 994 002 000 | 2 994,4 | 4 150 590,4 | 346,08 | 99,90 |
| | 3 500 | 3 994 002 000 | 3 495,8 | 4 417 540,9 | 742,18 | 99,89 |
| | 4 000 | 3 994 002 000 | 3 993,0 | 5 290 777,8 | 1 389,50 | 99,87 |

A Tabela 12 mostra que a formulação $\mathrm{PECR}_{\mathrm{ILP}}$ desconsidera um grande quantidade de restrições de transitividade (acima de $99\%$). Todas as instâncias são resolvidas em menos de 23 minutos. Esse resultado demonstra a escalabilidade dos modelos criados a partir da formulação $\mathrm{PECR}_{\mathrm{ILP}}$ no contexto dos grafos propostos por Bocker et al. [17].

## 6.5 CONCLUSÕES

Uma nova formulação, denotada aqui com $\mathrm{PECR}_{\mathrm{ILP}}$, foi proposta para o PEC. Os resultados experimentais mostram que ela é capaz de construir modelos matemáticos com um número de restrições de transitividade consideravelmente menor que os modelos criados a partir da formulação original. Os modelos criados pela formulação $\mathrm{PECR}_{\mathrm{ILP}}$ foram mais compactos (menos variáveis e restrições) e demandaram menos esforço computacional (tempo e memória) para serem solucionados de forma exata.

A nova formulação poderá ser combinada com outras técnicas, como os métodos de redução propostos por Bocker et al. [17]. Uma redução no tamanho das instâncias poderá prover ganhos ainda maiores em termos de tempo computacional e consumo de memória.

É esperado também que os resultados aqui apresentados, dentro do contexto do método exato, possam ser utilizados para criar novas heurísticas e técnicas de aproximação para o PEC.

# 7

# CONCLUSÃO

O agrupamento de dados é um problema importante em diferentes campos da ciência, e pode ser modelado através de um problema de particionamento de um grafo. Este trabalho estudou uma variante específica, denominada Particionamento de Grafos em Cliques (PGC), proposta inicialmente por Grotschel e Wakabayashi [42]. O objetivo do PGC é particionar os vértices de um grafo completo ponderado, obtendo-se uma partição formada por um conjunto de subgrafos completos que possuem um valor máximo para a soma do peso de suas arestas. Pode-se resolver esse problema de maneira exata através da formulação PLI proposta em [42]. Essa formulação, entretanto, cria modelos matemáticos com um grande número de restrições, que demandam um grande esforço computacional para serem solucionados.

Trabalhos recentes [27, 67, 9] propuseram estratégias para tentar a amenizar o esforço computacional necessário para solucionar o PGC de forma exata. Existem dois tipos de estratégias que podem ser utilizadas: criar uma nova formulação que crie modelos matemáticos mais compactos ou criar uma técnica de pré-processamento que reduza o tamanho do grafo que serve de entrada para esse problema. Esta tese explorou ambas as opções.

O desempenho apresentado pelas técnicas propostas nesta tese demonstram que a hipótese apresentada inicialmente é verdadeira. Os resultados apresentados nos Capítulos 4-6 demonstram que é possível desenvolver formulações que resultem em modelos matemáticos mais compactos do que os modelos criados a partir das formulações de [27, 67], e criar técnicas de pré-processamento do grafo do problema que consigam reduzir mais o grafo de entrada do problema do que a técnica apresentada em [9].

As principais contribuições deste trabalho foram apresentadas nos Capítulos 4, 5 e 6. No Capítulo 4, foram propostas duas técnicas que reduzem o tamanho do grafo do problema. Elas estendem os resultados de Arenas et al. [9], obtendo resultados superiores em termos de redução. Os modelos matemáticos criados a partir do grafo reduzido foram mais compactos (menos variáveis e restrições), e demandaram menos esforço computacional (tempo e memória) para serem solucionados de forma exata. O algoritmo de pré-processamento $\mathrm{BCCFilter}$, proposto nesta tese, se destacou frente aos outros algoritmos apresentados no capítulo em termos de complexidade assintótica e capacidade de reduzir o tamanho do problema.

O Capítulo 5 apresenta uma nova formulação, denominada $\mathrm{PGC}_{\mathrm{R3}}$, que cria modelos matemáticos mais compactos para o PGC aplicado ao contexto de agrupamento de dados qualitativos. Miyauchi e Sukegawa [67] identificaram uma condição suficiente que o peso das arestas dentro de um grupo de uma

partição ótima deve seguir para evitar que ele seja subdividido. Essa condição foi utilizada para propor uma formulação que cria modelos matemáticos mais compactos que a formulação original PGC. Nesta tese, uma nova condição para o peso das arestas dentro do grupo de uma partição ótima foi proposta. Essa condição foi utilizada para propor uma formulação que cria modelos matemáticos mais compactos ($\mathrm{PGC}_{\mathrm{R3}}$). Nos experimentos, os modelos criados a partir da formulação $\mathrm{PGC}_{\mathrm{R3}}$ são comparados com os modelos criados a partir das formulações propostas anteriormente na literatura. Os resultados experimentais mostraram que a formulação $\mathrm{PGC}_{\mathrm{R3}}$ se sobressaiu sobre as demais, criando modelos matemáticos mais compactos (menos variáveis e restrições), e que demandaram menos esforço computacional (tempo e memória) para serem solucionados de forma exata.

Uma nova formulação ($\mathrm{PECR}_{\mathrm{ILP}}$), capaz de criar modelos matemáticos mais compactos, foi proposta no Capítulo 6 para o PGC aplicado ao contexto do problema edição de *clusters*. Essa formulação foi derivada de um estudo que identificou quais restrições poderiam ser consideradas redundantes dentro do contexto desse problema. Esse estudou deu origem a uma nova formulação ($\mathrm{PECR}_{\mathrm{ILP}}$) que evita que restrições identificadas como redundantes sejam inseridas no modelo matemático. Nos experimentos, os modelos matemáticos criados a partir da formulação $\mathrm{PECR}_{\mathrm{ILP}}$ foram comparados com os modelos criados a partir da formulação original proposta para esse problema ($\mathrm{PEC}_{\mathrm{ILP}}$). Os resultados experimentais mostraram que a formulação $\mathrm{PECR}_{\mathrm{ILP}}$ se sobressaiu sobre a formulação $\mathrm{PEC}_{\mathrm{ILP}}$. Ela criou modelos matemáticos mais compactos (menos variáveis e restrições), e que demandaram menos esforço computacional (tempo e memória) para serem solucionados de forma exata.

Apesar dos resultados apresentados nessa tese, a redundância da formulação PLI proposta por Grotschel e Wakabayashi [42] continua a ser um desafio interessante a ser explorado. Formulações que resultem em modelos matemáticos ainda mais compactos ou técnicas de pré-processamento mais eficientes podem ser propostas a partir dos estudos realizados neste trabalho.

## 7.1 TRABALHOS FUTUROS

Durante o desenvolvimento da tese surgiram outras possibilidades de aplicação das técnicas propostas neste trabalho.

As formulações mais compactas e as técnicas de pré-processamento do grafo do problema propostas nesta tese, podem beneficiar outros algoritmos exatos. A diminuição da redundância da formulação do problema PGC, por exemplo, pode beneficiar diretamente o algoritmo de *Planos de Corte*, já que é necessário checar a viabilidade de todas as restrições de transitividade a cada iteração desse algoritmo. No contexto do algoritmo *Branch-and-Bound* a diminuição da redundância pode ser benéfica caso seja utilizada uma relaxação de programação linear a cada vértice da árvore de busca. O algoritmo de *Geração de Colunas* pode se beneficiar, por exemplo, do pré-processamento

do grafo $K_n$. Uma redução do tamanho desse grafo irá diminuir o número de variáveis da formulação do "problema mestre"[87].

Nas subseções que seguem serão descritas outras possibilidades de trabalhos futuros que apliquem os resultados dessa tese.

### 7.1.1 *Relaxação de Programação Linear do PGC*

Na relaxação de Programação Linear (PL), a restrição de integralidade $x_{ij} \in \{0,1\}$ é substituída por $x_{ij} \in [0,1]$ fornecendo um limitante superior à solução ótima do PGC. Esse limitante é frequente utilizado, por exemplo, por algoritmos de aproximação que utilizam a solução LP. O objetivo de um algoritmo de aproximação é tentar obter um valor próximo ao ótimo em um tempo pelo menos polinomial.

A eliminação de restrições redundantes pode beneficiar a relaxação PL do PGC. Dinh e Thai [27] propuseram uma formulação mais compacta para o problema de maximização de modularidade que mantém a solução ótima da relaxação PL desse problema. No mesmo artigo, foi proposto um algoritmo de aproximação baseado na relaxação PL. Os autores mostraram que o maior consumo de tempo computacional advém da solução da relaxação PL do problema. Com a formulação mais compacta, portanto, houve uma redução do tempo computacional do algoritmo de aproximação.

Como trabalho futuro, espera-se testar as relaxações PL das formulações mais compactas propostas neste trabalho. Elas podem ser utilizadas em algoritmos de aproximação para o problema de agrupamento de dados qualitativo (Capítulo 5) e o problema de edição de *clusters* (Capítulo 6). Baseado nos experimentos apresentados no trabalho de Dinh e Thai [27], é esperado um ganho em termos de tempo computacional para os algoritmos de aproximação.

### 7.1.2 *Propor uma formulação PGC mais compacta para modularidade*

Fazer um estudo dos triângulos que correspondem as restrições de transitividade na formulação PGC para o problema de maximização de modularidade e propor uma formulação mais compacta.

Pode-se encontrar dificuldades pois existem casos em que o peso das arestas do grafo do PGC ($K_n$), definido pela modularidade (Equação (7)), apresenta valores negativos mesmo existindo arestas entre os vértices no grafo do conjunto dados ($G$). No exemplo da Figura 32, o peso da aresta $\{33,34\}$ é $-0,0019$, entretanto, os vértices estão no mesmo grupo da partição ótima. Como essa aresta negativa irá formar triângulos com os demais vértices, a análise dos tipos de triângulos presentes em $K_n$ poderá ficar distorcida.

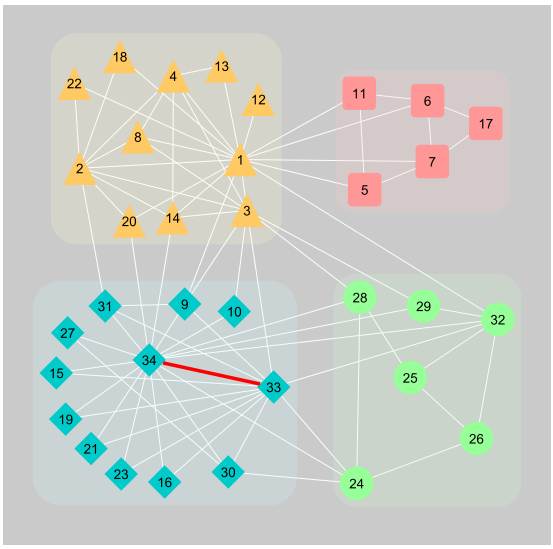



Figura 32: Partição ótima do conjunto de dados Zachary's karate club [89]. A aresta $\{33, 34\}$ foi destacada em vermelho.

### 7.1.3 *Propor uma formulação mais compacta para o PGC com restrições de capacidade*

No contexto do PGC com restrições de capacidade, o objetivo é particionar os vértices de um grafo completo $K_n = (V_n, E_m)$ em uma partição $\mathcal{P} = \{C_1, \ldots, C_n\}$ formada por subconjuntos de vértices (grupos), tal que a soma do peso das arestas dentro dos grupos seja máxima e o tamanho dos grupos estejam no intervalo $LB \leqslant |V_i| \leqslant UB$. Os parâmetros $LB$ e $UB$ representam, respectivamente, o limite inferior e superior para o tamanho dos grupos. O peso das arestas de $K_n$, assim como no PGC, é definido de acordo com o problema a ser resolvido.

Labbé e Özsoy [58] propuseram a seguinte formulação para esse problema:

$$(PGC_C)$$

$$\text{Maximizar} \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_{ij}$$

$$\text{sujeito a}$$

$$x_{ij} + x_{jk} - x_{ik} \leqslant 1 \quad , i < j < k \tag{21}$$

$$x_{ij} - x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k \tag{22}$$

$$-x_{ij} + x_{jk} + x_{ik} \leqslant 1 \quad , i < j < k \tag{23}$$

$$\sum_{j \in N_i} x_{ij} \geqslant LB - 1 \quad , i \in [1..n] \tag{24}$$

$$\sum_{j \in N_i} x_{ij} \leqslant UB - 1 \quad , i \in [1..n] \tag{25}$$

$$x_{ij} \in \{0, 1\} \quad , i < j$$

As variáveis $x_{ij}$ assumem o valor 1 quando os vértices $i$ e $j$ estão no mesmo grupo e $0$ caso contrário. As mesmas restrições de transitividade (21-23) da formulação PGC original são utilizadas nesse problema. Dois novos conjuntos de restrições de capacidade são inseridos para cada vértice do grafo $K_n$. Os limites $LB$ e $UB$ irão restringir a quantidade de vértices em cada grupo.

Pode-se resolver através dessa formulação o problema PGC com restrição mínima para o peso dos grupos. Basta utilizar um $UB = n$ e definir o tamanho mínimo dos grupos através de $LB$.

Pode-se notar que quando o limite $LB = 1$ e $UB = n$, a formulação $PGC_C$ coincide com a formulação $PGC$ não capacitada originalmente proposta por Grotschel e Wakabayashi [42].

A formulação $PGC_C$ apresenta as mesmas limitações do $PGC$, ou seja, um grande número de restrições de transitividade redundantes. A análise de triângulos aplicada nos Capítulos 5 e 6 pode ser utilizada para identificar tal redundância.

### 7.1.4 *Propor uma formulação mais compacta para o problema de Agregação de Rankings*

O problema de Agregação de *Rankings* (PAR) tem sido muito estudado em ciências sociais, onde o problema da junção de votos de múltiplos julgadores em uma eleição deve ser solucionado [11, 62].

Esse problema pode ser definido formalmente da seguinte maneira. Dado um conjunto de $n$ permutações, composta por $p$ elementos no intervalo $1, 2, ..., p$, procura-se determinar uma permutação que seja capaz de representar um consenso entre elas. A permutação consenso deve ser escolhida de forma a minimizar sua distância frente às demais permutações. A distância empregada é conhecida como *Kendall-tau* [54]. Ela mede o número de desacordos nas ordenações dos elementos de cada permutação.

O problema é semelhante ao problema de agrupamento de dados qualitativos apresentado no 5. A diferença se encontra na relação de ordem que o problema de ranking exige. Nesse caso, a propriedade de simetria ($x_{ij} = x_{ji}$) presente na formulação PLI do agrupamento de dados qualitativos não é válida para o PAR, pois a relação de ordem é antissimétrica. A formulação PLI para o PAR, foi definido em [22] como:

$$
\begin{aligned}
& (PAR) \\
\text{Maximizar} \quad & \sum_{i \neq j} w_{ij} x_{ij} \\
\text{sujeito a} \quad & \\
& x_{ij} + x_{jk} - x_{ik} \leqslant 1 \ , \ i \neq j \neq k \\
& x_{ij} + x_{ji} = 1 \quad , \ i \neq j \\
& x_{ij} \in \{0, 1\} \quad , \ i \neq j, \quad i, j \in [1..n]
\end{aligned}
$$

em que $x_{ij} = 1$ se $i$ está posicionado antes de $j$ no ranking, e $x_{ij} = 0$ caso contrário. O valor de $w_{ij}$ representa o número de permutações em que $i$ está posicionado antes de $j$ menos o número de permutações em que $j$ está posicionado antes de $i$.

Este problema possui as mesmas limitações que o problema de agrupamento de dados qualitativos apresentado no Capítulo 5. Os modelos PLI apresenta um número elevado de restrições de transitividade. Pode-se, portanto, realizar um estudo dos triângulos referentes as restrições de transitividade desse problema para definir cláusulas condicionais que possam eliminar restrições redundantes.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] AGARWAL, G.; KEMPE, D. Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*, v. 66, n. 3, p. 409–418, 2008.
Disponível em http://doi.acm.org/10.1140/epjb/e2008-00425-1

[2] AGGARWAL, C. C.; REDDY, C. K. *Data clustering: Algorithms and applications*. 1st ed. Chapman & Hall/CRC, 2013.
Disponível em https://books.google.com.br/books?isbn=1498785778

[3] AILON, N.; CHARIKAR, M.; NEWMAN, A. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, v. 55, n. 5, p. 23:1–23:27, 2008.
Disponível em http://doi.acm.org/10.1145/1411509.1411513

[4] ALBERT, R.; BARABÁSI, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics*, v. 74, n. 1, p. 47, 2002.
Disponível em https://doi.org/10.1103/RevModPhys.74.47

[5] ALOISE, D.; CAFIERI, S.; CAPOROSSI, G.; HANSEN, P.; PERRON, S.; LIBERTI, L. Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, v. 82, n. 4, p. 046112, 2010.
Disponível em https://doi.org/10.1103/PhysRevE.82.046112

[6] ALUSH, A.; GOLDBERGER, J. Ensemble segmentation using efficient integer linear programming. *IEEE transactions on pattern analysis and machine intelligence*, v. 34, n. 10, p. 1966–1977, 2012.
Disponível em https://doi.org/10.1109/TPAMI.2011.280

[7] ANDRES, B.; KAPPES, J. H.; BEIER, T.; KÖTHE, U.; HAMPRECHT, F. A. Probabilistic image segmentation with closedness constraints. In: *2011 International Conference on Computer Vision*, IEEE, 2011, p. 2611–2618.
Disponível em https://doi.org/10.1109/ICCV.2011.6126550

[8] ANDREWS, R. L.; BRUSCO, M. J.; CURRIM, I. S. Amalgamation of partitions from multiple segmentation bases: A comparison of non-model-based and model-based methods. *European Journal of Operational Research*, v. 201, n. 2, p. 608–618, 2010.
Disponível em https://doi.org/10.1016/j.ejor.2009.03.002

[9] ARENAS, A.; DUCH, J.; FERNÁNDEZ, A.; GÓMEZ, S. Size reduction of complex networks preserving modularity. *New Journal of Physics*, v. 9, n. 6, p. 176, 2007.
Disponível em https://doi.org/10.1088/1367-2630/9/6/176

[10] BANSAL, N.; BLUM, A.; CHAWLA, S. Correlation clustering. *Machine Learning*, v. 56, n. 1, p. 89–113, 2004.
Disponível em https://doi.org/10.1023/B:MACH.0000033116.57574.95

[11] BARTHOLDI, J., I.; TOVEY, C.; TRICK, M. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, v. 6, n. 2, p. 157–165, 1989.
Disponível em http://dx.doi.org/10.1007/BF00303169

[12] BASTOS, L.; OCHI, L. S.; PROTTI, F.; SUBRAMANIAN, A.; MARTINS, I. C.; PINHEIRO, R. G. S. Efficient algorithms for cluster editing. *Journal of Combinatorial Optimization*, v. 31, n. 1, p. 347–371, 2016.
Disponível em https://doi.org/10.1007/s10878-014-9756-7

[13] BEN-DOR, A.; SHAMIR, R.; YAKHINI, Z. Clustering gene expression patterns. *Journal of computational biology*, v. 6, n. 3-4, p. 281–297, 1999.
Disponível em https://doi.org/10.1089/106652799318274

[14] BERGMANN, B.; HOMMEL, G. Improvements of general multiple test procedures for redundant systems of hypotheses. In: *Multiple Hypothesenprüfung/Multiple Hypotheses Testing*, Springer, p. 100–115, 1988.
Disponível em https://doi.org/10.1007/978-3-642-52307-6_8

[15] BÖCKER, S.; BAUMBACH, J. Cluster editing. In: BONIZZONI, P.; BRATTKA, V.; LÖWE, B., eds. *The Nature of Computation. Logic, Algorithms, Applications*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, p. 33–44.
Disponível em https://link.springer.com/book/10.1007/978-3-642-39053-1

[16] BÖCKER, S.; BRIESEMEISTER, S.; BUI, Q. B. A.; TRUSS, A. Going weighted: Parameterized algorithms for cluster editing. *Theoretical Computer Science*, v. 410, n. 52, p. 5467–5480, 2009.
Disponível em https://doi.org/10.1016/j.tcs.2009.05.006

[17] BÖCKER, S.; BRIESEMEISTER, S.; KLAU, G. W. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, v. 60, n. 2, p. 316–334, 2011.
Disponível em https://doi.org/10.1007/s00453-009-9339-7

[18] BONCHI, F.; MORALES, G. D. F.; GIONIS, A.; UKKONEN, A. Activity preserving graph simplification. *Data mining and knowledge discovery*, v. 27, n. 3, p. 321–343, 2013.
Disponível em https://doi.org/10.1007/s10618-013-0328-8

[19] BRUSCO, M. J.; KÖHN, H.-F. Clustering qualitative data based on binary equivalence relations: neighborhood search heuristics for the clique parti-

tioning problem. *Psychometrika*, v. 74, n. 4, p. 685, 2009.
Disponível em https://doi.org/10.1007/s11336-009-9126-z

[20] CALVO, B.; SANTAFE, G. scmamp: Statistical comparison of multiple algorithms in multiple problems. 2015.
Disponível em https://cran.r-project.org/web/packages/scmamp/index.html

[21] CHARIKAR, M.; GURUSWAMI, V.; WIRTH, A. Clustering with qualitative information. *Journal of Computer and System Sciences*, v. 71, n. 3, p. 360 – 383, learning Theory 2003, 2005.
Disponível em https://doi.org/10.1016/j.jcss.2004.10.012

[22] CHARON, I.; HUDRY, O. An updated survey on the linear ordering problem for weighted or unweighted tournaments. *Annals of Operations Research*, v. 175, n. 1, p. 107–158, 2010.
Disponível em https://doi.org/10.1007/s10479-009-0648-7

[23] CHEN, Z.; YANG, S.-H.; XU, H.; LI, J.-J. Abo blood group system and the coronary artery disease: an updated systematic review and meta-analysis. *Scientific reports*, v. 6, 2016.
Disponível em https://doi.org/10.1038/srep23250

[24] DE AMORIM, S. G.; BARTHÉLEMY, J.-P.; RIBEIRO, C. C. Clustering and clique partitioning: simulated annealing and tabu search approaches. *Journal of Classification*, v. 9, n. 1, p. 17–41, 1992.
Disponível em https://doi.org/10.1007/BF02618466

[25] DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, v. 7, n. Jan, p. 1–30, 2006.
Disponível em http://www.jmlr.org/papers/v7/demsar06a.html

[26] DIESTEL, R. *Graph theory*. Electronic library of mathematics. Springer, 2017.
Disponível em https://www.springer.com/la/book/9783662536216

[27] DINH, T. N.; THAI, M. T. Finding community structure with performance guarantees in complex networks. *arXiv preprint arXiv:1108.4034*, 2011.
Disponível em https://arxiv.org/abs/1108.4034

[28] DINH, T. N.; THAI, M. T. Toward optimal community detection: From trees to general weighted networks. *Internet Mathematics*, v. 11, n. 3, 2015.
Disponível em https://doi.org/10.1080/15427951.2014.950875

[29] DORNDORF, U.; JAEHN, F.; PESCH, E. Modelling robust flight-gate scheduling as a clique partitioning problem. *Transportation Science*, v. 42,

n. 3, p. 292–301, 2008.
Disponível em https://doi.org/10.1287/trsc.1070.0211

[30] DORNDORF, U.; JAEHN, F.; PESCH, E. Flight gate assignment and recovery strategies with stochastic arrival and departure times. *OR spectrum*, v. 39, n. 1, p. 65–93, 2017.
Disponível em https://doi.org/10.1007/s00291-016-0443-1

[31] DORNDORF, U.; PESCH, E. Fast clustering algorithms. *ORSA Journal on Computing*, v. 6, n. 2, p. 141–153, 1994.
Disponível em https://doi.org/10.1287/ijoc.6.2.141

[32] FELLOWS, M.; LANGSTON, M.; ROSAMOND, F.; SHAW, P. Efficient parameterized preprocessing for cluster editing. In: CSUHAJ-VARJÚ, E.; ÉSIK, Z., eds. *Fundamentals of Computation Theory*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, p. 312–321.
Disponível em https://doi.org/10.1007/978-3-540-74240-1_27

[33] FORTUNATO, S. Community detection in graphs. *Physics reports*, v. 486, n. 3-5, p. 75–174, 2010.
Disponível em https://doi.org/10.1016/j.physrep.2009.11.002

[34] FORTUNATO, S.; HRIC, D. Community detection in networks: A user guide. *Physics Reports*, v. 659, p. 1–44, 2016.
Disponível em https://doi.org/10.1016/j.physrep.2016.09.002

[35] FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, v. 32, n. 200, p. 675–701, 1937.
Disponível em https://doi.org/10.2307/2279372

[36] GARCÍA, S.; FERNÁNDEZ, A.; LUENGO, J.; HERRERA, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, v. 180, n. 10, p. 2044–2064, 2010.
Disponível em https://doi.org/10.1016/j.ins.2009.12.010

[37] GARCIA, S.; HERRERA, F. An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. *Journal of Machine Learning Research*, v. 9, n. Dec, p. 2677–2694, 2008.
Disponível em http://www.jmlr.org/papers/v9/garcia08a.html

[38] GEMMETTO, V.; CARDILLO, A.; GARLASCHELLI, D. Irreducible network backbones: unbiased graph filtering via maximum entropy. *arXiv preprint arXiv:1706.00230*, 2017.
Disponível em https://arxiv.org/abs/1706.00230

[39] GIONIS, A.; MANNILA, H.; TSAPARAS, P. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, v. 1, n. 1, p. 4, 2007.
Disponível em https://doi.org/10.1145/1217299.1217303

[40] GIONIS, A.; ROZENSHTEIN, P.; TATTI, N.; TERZI, E. Community-aware network sparsification. In: *Proceedings of the 2017 SIAM International Conference on Data Mining*, SIAM, 2017, p. 426–434.
Disponível em https://doi.org/10.1137/1.9781611974973.48

[41] GOEKE, C.; KORNPETPANEE, S.; KÖSTER, M.; FERNÁNDEZ-REVELLES, A. B.; GRAMANN, K.; KÖNIG, P. Cultural background shapes spatial reference frame proclivity. *Scientific reports*, v. 5, 2015.
Disponível em https://doi.org/10.1038/srep11426

[42] GRÖTSCHEL, M.; WAKABAYASHI, Y. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, v. 45, n. 1, p. 59–96, 1989.
Disponível em https://doi.org/10.1007/BF01589097

[43] GUO, J. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, v. 410, n. 8, p. 718 – 726, 2009.
Disponível em https://doi.org/10.1016/j.tcs.2008.10.021

[44] HAGHTALAB, S.; XANTHOPOULOS, P.; MADANI, K. A robust unsupervised consensus control chart pattern recognition framework. *Expert Systems with Applications*, v. 42, n. 19, p. 6767–6776, 2015.
Disponível em https://doi.org/10.1016/j.eswa.2015.04.069

[45] HODGES, J.; LEHMANN, E. L.; *et al.* Rank methods for combination of independent experiments in analysis of variance. *The Annals of Mathematical Statistics*, v. 33, n. 2, p. 482–497, 1962.
Disponível em https://projecteuclid.org/euclid.aoms/1177704575

[46] HOPCROFT, J.; TARJAN, R. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, v. 16, n. 6, p. 372–378, 1973.
Disponível em https://doi.org/10.1145/362248.362272

[47] HÜFFNER, F.; KOMUSIEWICZ, C.; LIEBTRAU, A.; NIEDERMEIER, R. Partitioning biological networks into highly connected clusters with maximum edge coverage. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, v. 11, n. 3, p. 455–467, 2014.
Disponível em https://doi.org/10.1109/TCBB.2013.177

[48] IBM Ibm ilog cplex 12.7.1. 1987-2017.
Disponível em https://www.ibm.com/analytics/cplex-optimizer

[49] IMAN, R. L.; DAVENPORT, J. M. Approximations of the critical region of the friedman statistic. *Communications in Statistics-Theory and Methods*,

v. 9, n. 6, p. 571–595, 1980.
Disponível em https://doi.org/10.1080/03610928008827904

[50] IZUNAGA, Y.; YAMAMOTO, Y. A cutting plane algorithm for modularity maximization problem. *Journal of the Operations Research Society of Japan*, v. 60, n. 1, p. 24–42, 2017.
Disponível em https://doi.org/10.15807/jorsj.60.24

[51] JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, v. 31, n. 8, p. 651–666, 2010.
Disponível em https://doi.org/10.1016/j.patrec.2009.09.011

[52] KAPPES, J. H.; SPETH, M.; ANDRES, B.; REINELT, G.; SCHN, C. Globally optimal image partitioning by multicuts. In: *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, Springer, 2011, p. 31–44.
Disponível em https://doi.org/10.1007/978-3-642-23094-3_3

[53] KAPPES, J. H.; SPETH, M.; REINELT, G.; SCHNÖRR, C. Higher-order segmentation via multicuts. *Computer Vision and Image Understanding*, v. 143, p. 104–119, 2016.
Disponível em https://doi.org/10.1016/j.cviu.2015.11.005

[54] KENDALL, M. G. Rank correlation methods. 1948.
Disponível em https://doi.org/10.1111/j.2044-8317.1956.tb00172.x

[55] KIM, J.-R.; KIM, J.; KWON, Y.-K.; LEE, H.-Y.; HESLOP-HARRISON, P.; CHO, K.-H. Reduction of complex signaling networks to a representative kernel. *Science signaling*, v. 4, n. 175, p. ra35–ra35, 2011.
Disponível em https://doi.org/10.1126/scisignal.2001390

[56] KIM, S.; NOWOZIN, S.; KOHLI, P.; YOO, C. D. Higher-order correlation clustering for image segmentation. In: *Advances in neural information processing systems*, 2011, p. 1530–1538.
Disponível em url={http://dl.acm.org/citation.cfm?id=2986459.2986630}

[57] KOCHENBERGER, G.; GLOVER, F.; ALIDAEE, B.; WANG, H. Clustering of microarray data via clique partitioning. *Journal of Combinatorial Optimization*, v. 10, n. 1, p. 77–92, 2005.
Disponível em https://doi.org/10.1007/s10878-005-1861-1

[58] LABBÉ, M.; ÖZSOY, F. A. Size-constrained graph partitioning polytopes. *Discrete mathematics*, v. 310, n. 24, p. 3473–3493, 2010.
Disponível em https://doi.org/10.1016/j.disc.2010.08.009

[59] LANCICHINETTI, A.; FORTUNATO, S.; RADICCHI, F. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, v. 78,

p. 046110, 2008.
Disponível em https://link.aps.org/doi/10.1103/PhysRevE.78.046110

[60] LI, C.; SUN, L.; JIA, J.; CAI, Y.; WANG, X. Risk assessment of water pollution sources based on an integrated k-means clustering and set pair analysis method in the region of shiyan, china. *Science of The Total Environment*, v. 557, p. 307–316, 2016.
Disponível em https://doi.org/10.1016/j.scitotenv.2016.03.069

[61] LICHMAN, M. UCI machine learning repository. 2013.
Disponível em http://archive.ics.uci.edu/ml

[62] LORENA, L. H. N.; LORENA, A. C.; LORENA, L. A. N.; DE LEON FERREIRA, A. C. P.; *et al.* Clustering search applied to rank aggregation. In: *Intelligent Systems (BRACIS), 2014 Brazilian Conference on*, IEEE, 2014, p. 198–203.
Disponível em https://doi.org/10.1109/BRACIS.2014.44

[63] LORENA, L. H. N.; QUILES, M. G.; DE LEON FERREIRA, A. C. P.; LORENA, L. A. N. Preprocessing technique for cluster editing via integer linear programming. In: *International Conference on Intelligent Computing (ICIC)*, Springer, 2018, p. 287–297.
Disponível em https://doi.org/10.1007/978-3-319-95930-6_27

[64] LORENA, L. H. N.; QUILES, M. G.; LORENA, L. A. N. Improving the performance of an integer linear programming community detection algorithm trough clique filtering. In: *2019 19th International Conference on Computational Science and Applications (ICCSA)*, IEEE, 2019, p. to appear.

[65] LORENA, L. H. N.; QUILES, M. G.; LORENA, L. A. N.; DE CARVALHO, A. C. P. L. F.; CESPEDES, J. G. Qualitative data clustering: a new integer linear programming model. In: *The 2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, p. to appear.

[66] MIYAUCHI, A.; MIYAMOTO, Y. Computing an upper bound of modularity. *The European Physical Journal B*, v. 86, n. 7, p. 1–7, 2013.
Disponível em https://doi.org/10.1140/epjb/e2013-40006-7

[67] MIYAUCHI, A.; SUKEGAWA, N. Redundant constraints in the standard formulation for the clique partitioning problem. *Optimization Letters*, v. 9, n. 1, p. 199–207, 2015.
Disponível em https://doi.org/10.1007/s11590-014-0754-6

[68] NEWMAN, M. E. The structure and function of complex networks. *SIAM review*, v. 45, n. 2, p. 167–256, 2003.
Disponível em https://doi.org/10.1137/S003614450342480

[69] NEWMAN, M. E. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, v. 103, n. 23, p. 8577–8582, 2006.
Disponível em https://doi.org/10.1073/pnas.0601602103

[70] NEWMAN, M. E. J.; GIRVAN, M. Finding and evaluating community structure in networks. *Phys. Rev. E*, v. 69, p. 026113, 2004.
Disponível em https://doi.org/10.1103/PhysRevE.69.026113

[71] NOWOZIN, S.; KOHLI, P.; YOO, C.; KIM, S. Image segmentation using higher-order correlation clustering. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, , n. 1, p. 1, 2014.
Disponível em https://doi.org/10.1109/TPAMI.2014.2303095

[72] OOSTEN, M.; RUTTEN, J. H.; SPIEKSMA, F. C. The clique partitioning problem: facets and patching facets. *Networks*, v. 38, n. 4, p. 209–226, 2001.
Disponível em https://doi.org/10.1002/net.10004

[73] PALUBECKIS, G.; OSTREIKA, A.; TOMKEVIČIUS, A. An iterated tabu search approach for the clique partitioning problem. *The Scientific World Journal*, v. 2014, 2014.
Disponível em http://dx.doi.org/10.1155/2014/353101

[74] PROTTI, F.; DANTAS DA SILVA, M.; SZWARCFITER, J. L. Applying modular decomposition to parameterized cluster editing problems. *Theory of Computing Systems*, v. 44, n. 1, p. 91–104, 2009.
Disponível em https://doi.org/10.1007/s00224-007-9032-7

[75] RAMBAU, J.; SCHWARZ, C. A new branch and bound algorithm for the clique partitioning problem. In: *Operations Research Proceedings 2008*, Springer, p. 457–462, 2009.
Disponível em https://doi.org/10.1007/978-3-642-00142-0_74

[76] ROSSI, R. A.; AHMED, N. K. The network data repository with interactive graph analytics and visualization. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
Disponível em http://networkrepository.com

[77] SCHANK, T.; WAGNER, D. Finding, counting and listing all triangles in large graphs, an experimental study. In: *International Workshop on Experimental and Efficient Algorithms*, Springer, 2005, p. 606–609.
Disponível em https://doi.org/10.1007/11427186_54

[78] SHAMIR, R.; SHARAN, R.; TSUR, D. Cluster graph modification problems. In: GOOS, G.; HARTMANIS, J.; VAN LEEUWEN, J.; KUČERA, L., eds. *Graph-Theoretic Concepts in Computer Science*, Berlin, Heidelberg:

Springer Berlin Heidelberg, 2002, p. 379–390.
Disponível em https://doi.org/10.1007/3-540-36379-3_33

[79] SPEARMAN, C. The proof and measurement of association between two things. *The American Journal of Psychology*, v. 15, n. 1, p. 72–101, 1904.
Disponível em https://doi.org/10.2307/1422689

[80] STANLEY, N.; KWITT, R.; NIETHAMMER, M.; MUCHA, P. J. Compressing networks with super nodes. *Scientific reports*, v. 8, n. 1, p. 10892, 2018.
Disponível em https://doi.org/10.1038/s41598-018-29174-3

[81] STROGATZ, S. H. Exploring complex networks. *Nature*, v. 410, n. 6825, p. 268–276, 2001.
Disponível em https://doi.org/10.1038/35065725

[82] TAUER, G.; DATE, K.; NAGI, R.; SUDIT, M. An incremental graph-partitioning algorithm for entity resolution. *Information Fusion*, v. 46, p. 171–183, 2019.
Disponível em https://doi.org/10.1016/j.inffus.2018.06.001

[83] TEAM, R. D. C. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0, 2008.
Disponível em http://www.R-project.org

[84] WANG, H.; ALIDAEE, B.; GLOVER, F.; KOCHENBERGER, G. Solving group technology problems via clique partitioning. *International Journal of Flexible Manufacturing Systems*, v. 18, n. 2, p. 77–97, 2006.
Disponível em https://doi.org/10.1007/s10696-006-9011-3

[85] WANG, H.; OBREMSKI, T.; ALIDAEE, B.; KOCHENBERGER, G. Clique partitioning for clustering: a comparison with k-means and latent class analysis. *Communications in Statistics-Simulation and Computation*, v. 37, n. 1, p. 1–13, 2007.
Disponível em https://doi.org/10.1080/03610910701723559

[86] WATTS, D. J.; STROGATZ, S. H. Collective dynamics of 'small-world' networks. *nature*, v. 393, n. 6684, p. 440–442, 1998.
Disponível em https://doi.org/10.1038/30918

[87] WOLSEY, L. *Integer programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998.
Disponível em https://books.google.com.br/books?isbn=9780471283669

[88] XIAO, Y.; MACARTHUR, B. D.; WANG, H.; XIONG, M.; WANG, W. Network quotients: Structural skeletons of complex systems. *Physical*

*Review E*, v. 78, n. 4, p. 046102, 2008.
Disponível em https://doi.org/10.1103/physreve.78.046102

[89] ZACHARY, W. W. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, v. 33, n. 4, p. 452–473, 1977.
Disponível em https://www.jstor.org/stable/3629752

[90] ZHOU, Y.; HAO, J.-K.; GOËFFON, A. A three-phased local search approach for the clique partitioning problem. *Journal of Combinatorial Optimization*, v. 32, n. 2, p. 469–491, 2016.
Disponível em https://doi.org/10.1007/s10878-015-9964-9